NASA CR-172088

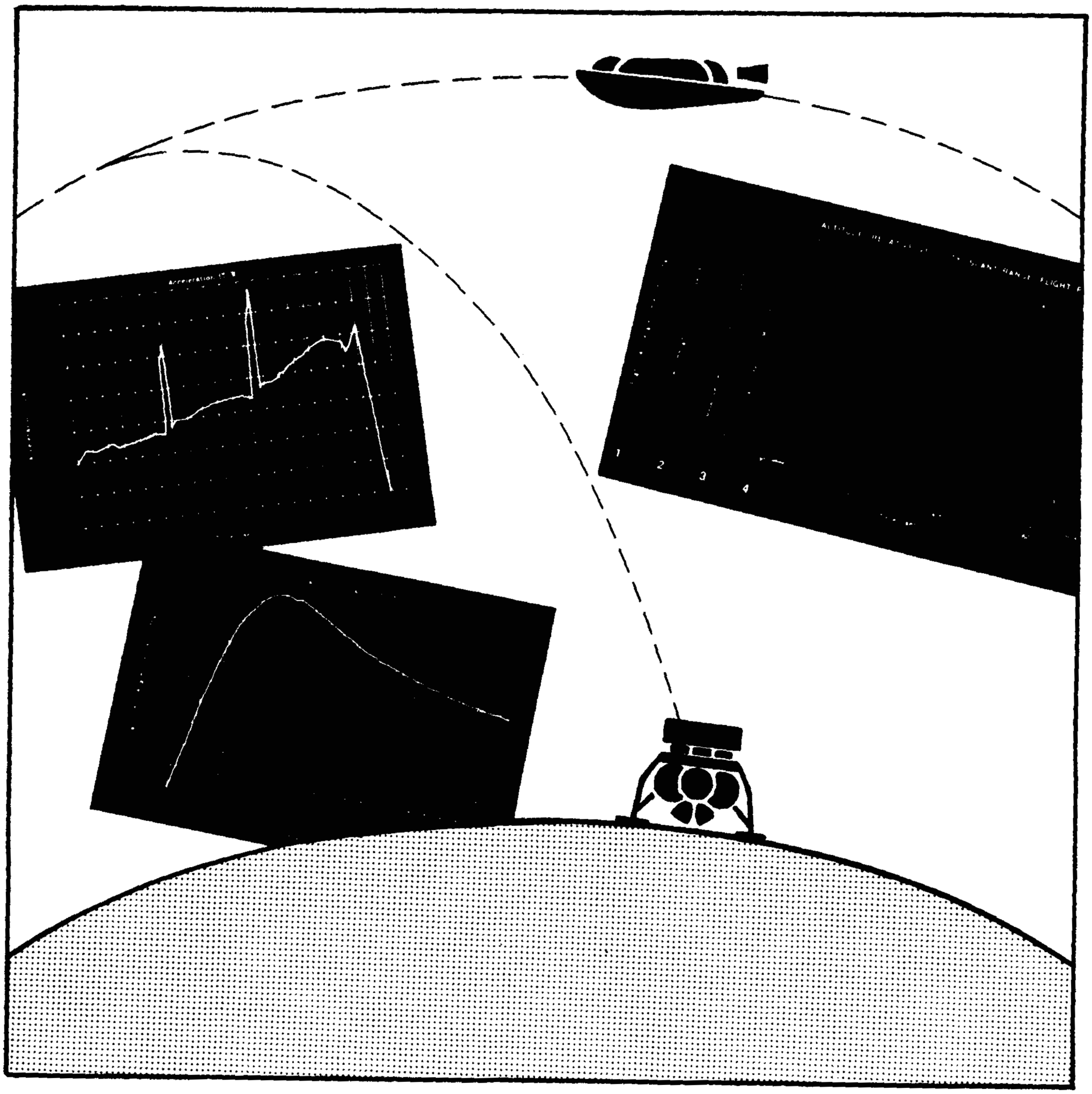
(NASA-CR-172088) LANDER PROGRAM MANUAL: A LUNAR ASCENT AND DESCENT SIMULATION (Eagle Engineering) 88 p CSCL 22B

N89-15157

Unclas
G3/18 0179854

LBSS

Lander Program Manual



NASA Contract Number NAS9-17878
EEI Report 88-195
September 30, 1988

EAGLE

LANDER
Program Manual

A Lunar
Ascent and Descent
Simulation

Prepared for the
National Aeronautics and Space Administration
Johnson Space Center
Advanced Programs Office
as part of the
Advanced Space Transportation Support Contract (ASTS)
and the
Lunar Base Systems Study (LBSS)

Contract Number: NAS 9-17878

by
Eagle Engineering, Incorporated
Report Number: 88-195

September 30, 1988

FOREWORD

This report documents the first edition of LANDER, a lunar ascent and descent trajectory simulation program. The purpose of the program is to provide delta velocity and trajectory information for lunar ascent and descent between low lunar orbit and the lunar surface. This information will aid in the formulation of plans to return to the Moon.

Dr. J.W. Alred was the NASA technical monitor for the ASTS contract. Mr. A. Petro was the NASA task monitor for this activity. The Eagle project manager was Mr. W.R. Stump. Special thanks go to Mr. J. Funk for his helpful advice and valuable assistance. This program was written and documented by Mr. C.C. Varner.

TABLE OF CONTENTS

		Page
Foreword		ii
Table of Contents		iii
List of Figures		iv
List of Tables		v
1.0	Introduction	1
2.0	Program Operation	2
3.0	Data Entry Subroutine	5
4.0	Variable Initialization	13
5.0	Integration Subroutine	15
6.0	Equations of Motion	19
	6.1 Control Procedures	23
	6.2 Thrust Profile	25
7.0	Output Subroutine	26
8.0	Orbit Subroutine	28
Appendix A -- List of Variables and Arrays		32
Appendix B -- Program Listings		35
Appendix C -- Examples of Output		73

LIST OF FIGURES

		Page
Figure 1 :	Main Program Flow Chart	3
Figure 2 :	Orbital Elements	4
Figure 3 :	Data Entry Flow Chart	6
Figure 4 :	Approach Paths for Landing	11
Figure 5 :	The Approach Azimuth	12
Figure 6 :	Heading, Azimuth, and Inclination	13
Figure 7 :	Variable Initialization Flow Chart	15
Figure 8 :	Integrator Flow Chart	18
Figure 9 :	Equations of Motion	21
Figure 10:	Spherical Coordinates	23
Figure 11:	Thrust Pitch Angle	24
Figure 12:	Control Model	25
Figure 13:	Lander Thrust Profile	26
Figure 14:	Output Flow Chart	28
Figure 15:	Orbit Flow Chart	30
Figure 16:	Radial Coordinates	31

LIST OF TABLES

		Page
Table 1:	Calculation of the Pseudo-inclination	10
Table 2:	Heading and Azimuth Relationship to Pseudo-inclination	11
Table A1:	Variable Arrays	33
Table A2:	Variables	33

1.0 INTRODUCTION

LANDER is a computer program used to predict the trajectory and flight performance of a spacecraft ascending or descending between a low lunar orbit of 15 to 500 nautical miles (nm) and the lunar surface. It is a three degree-of-freedom simulation which is used to analyze the translational motion of the vehicle during descent. Attitude dynamics and rotational motion are not considered.

The program can be used to simulate either an ascent from the Moon or a descent to the Moon. For an ascent, the spacecraft is initialized at the lunar surface and accelerates vertically away from the ground at full thrust. When the local velocity becomes 30 ft/s, the vehicle turns downrange with a pitch-over maneuver and proceeds to fly a gravity turn until Main Engine Cut-off (MECO). The spacecraft then coasts until it reaches the requested holding orbit where it performs an orbital insertion burn.

During a descent simulation, the lander begins in the holding orbit and performs a deorbit burn. It then coasts to pericynthion, where it reignites its engines and begins a gravity turn descent. When the local horizontal velocity becomes zero, the lander pitches up to a vertical orientation and begins to hover in search of a landing site. The lander hovers for a period of time specified by the user, and then lands.

Newton-Raphson iteration techniques are used to optimize the pitch-over maneuver and the MECO time for proper orbit insertion. Integration is performed using a Runge-Kutta fourth order integrator. This integrator has been verified with launch simulations of the Titan and Conestoga launch vehicles. LANDER receives input, presents output, and does all calculations in English units. The basic coordinate system is spherical. The moon is modelled as a spherical body of uniform gravity having no atmosphere and no gravitational harmonics.

Even though the output for a descent simulation appears to start at orbit and end at the surface, the mathematical calculations are performed in reverse. The program actually initializes the lander at the lunar surface and proceeds to simulate an ascent using negative mass flow. After the proper orbit has been achieved the data is reorganized and printed in the proper chronological sequence for a descent. Note: that this "reversed flight" is only characteristic of the descent simulations.

2.0 PROGRAM OPERATION

LANDER has a main driver program which accesses nine subroutines, and two function routines. In FORTRAN versions, the function routines are not necessary; the FORTRAN language has standard functions which perform the same operations. The main program controls the flow of operations to and from the subroutines. The subroutines perform activities such as input, output, and analytical calculations. The function routines perform basic numerical or mathematical calculations. A flow chart of the main program is shown in Figure 1.

The program must first define the functions and dimension the arrays that are to be used. BASIC versions of LANDER have a function equation for the ArcCosine since it is not an intrinsic function within this programming language. This function equation has some singularities which are corrected using tests within the ArcCosine function routine at lines 22000-22360. The ArcTangent 360 function routine between lines 21000 and 21250 is also necessary in BASIC versions of LANDER since the intrinsic ArcTangent function does not test for quadrant.

In the Data Entry subroutine, the user enters (or tells the program where to find) data that is used during the simulation. The following information is necessary for operation.

- The longitude and latitude of the landing site
- The weight of the payload to be carried
- The rocket characteristics such as maximum thrust level,
  specific impulse, inert weight, and propellant weight
- The amount of time that the lander is expected to hover before landing
- The holding orbit apocynthion and pericynthion
- The holding orbit inclination
- The estimated pitch-over angle
- The estimated main engine cut-off time

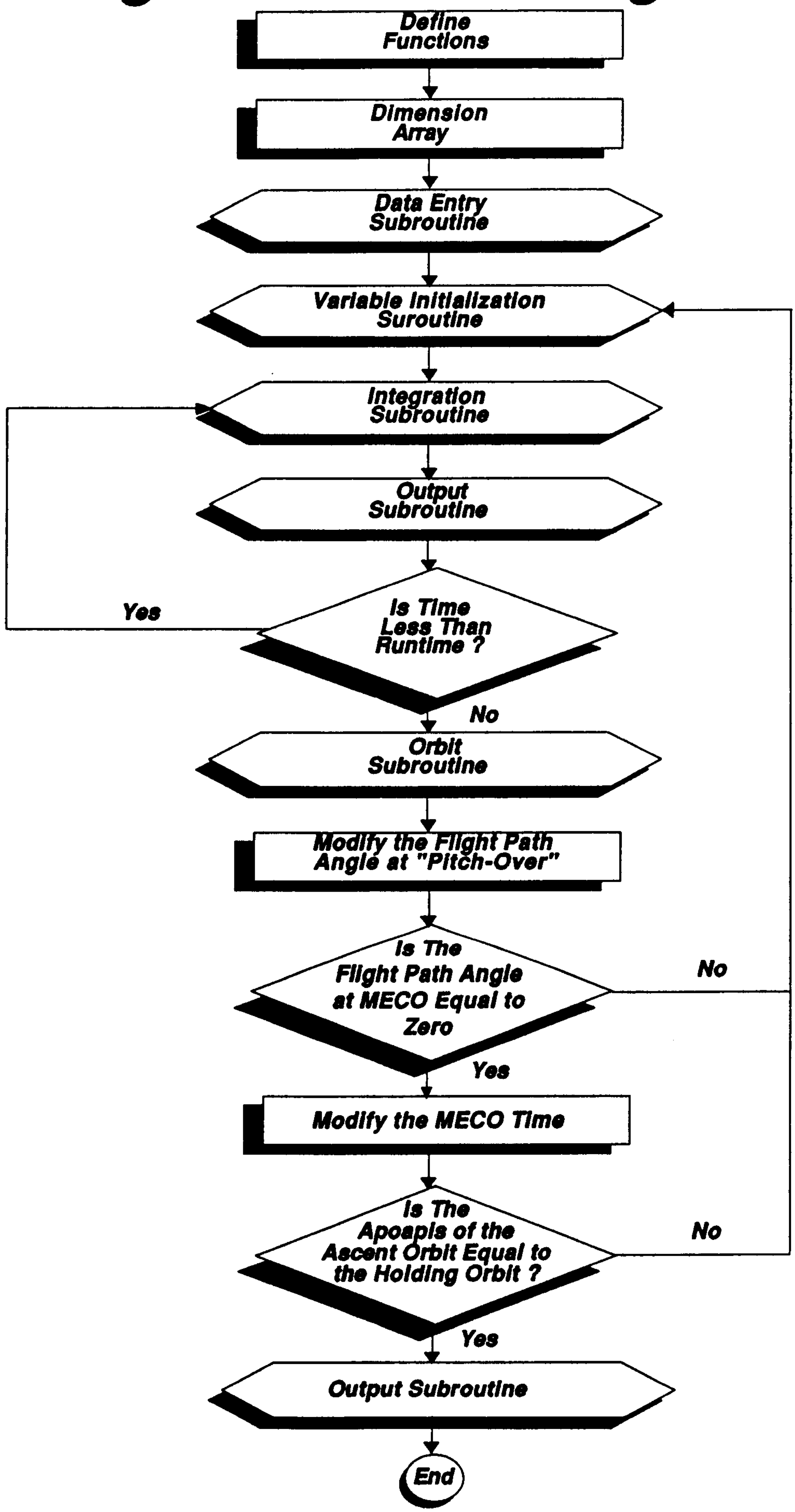
The Data Entry subroutine returns this information to the main program which immediately transfers control to the Variable Initialization subroutine.

A trajectory may be run numerous times during a simulation. Each time, it must start with the same initial conditions. The Variable Initialization subroutine sets all of the preliminary variables used during the integration calculations.

The Integration subroutine uses the state vector and the Equations of Motion to determine a new state vector at a future time (1 second later). The integration technique is a 4th order Runge-Kutta method, which makes four estimates of how the state vector changes during the time step. These estimates are then weighted and averaged to obtain a state vector change which has fourth order accuracy (0.01%).

Optimal step size control is not utilized. The time step of one second is fixed for the duration of program execution. This simplifies output operations at the expense of integration time efficiency.

Figure 1: Main Program

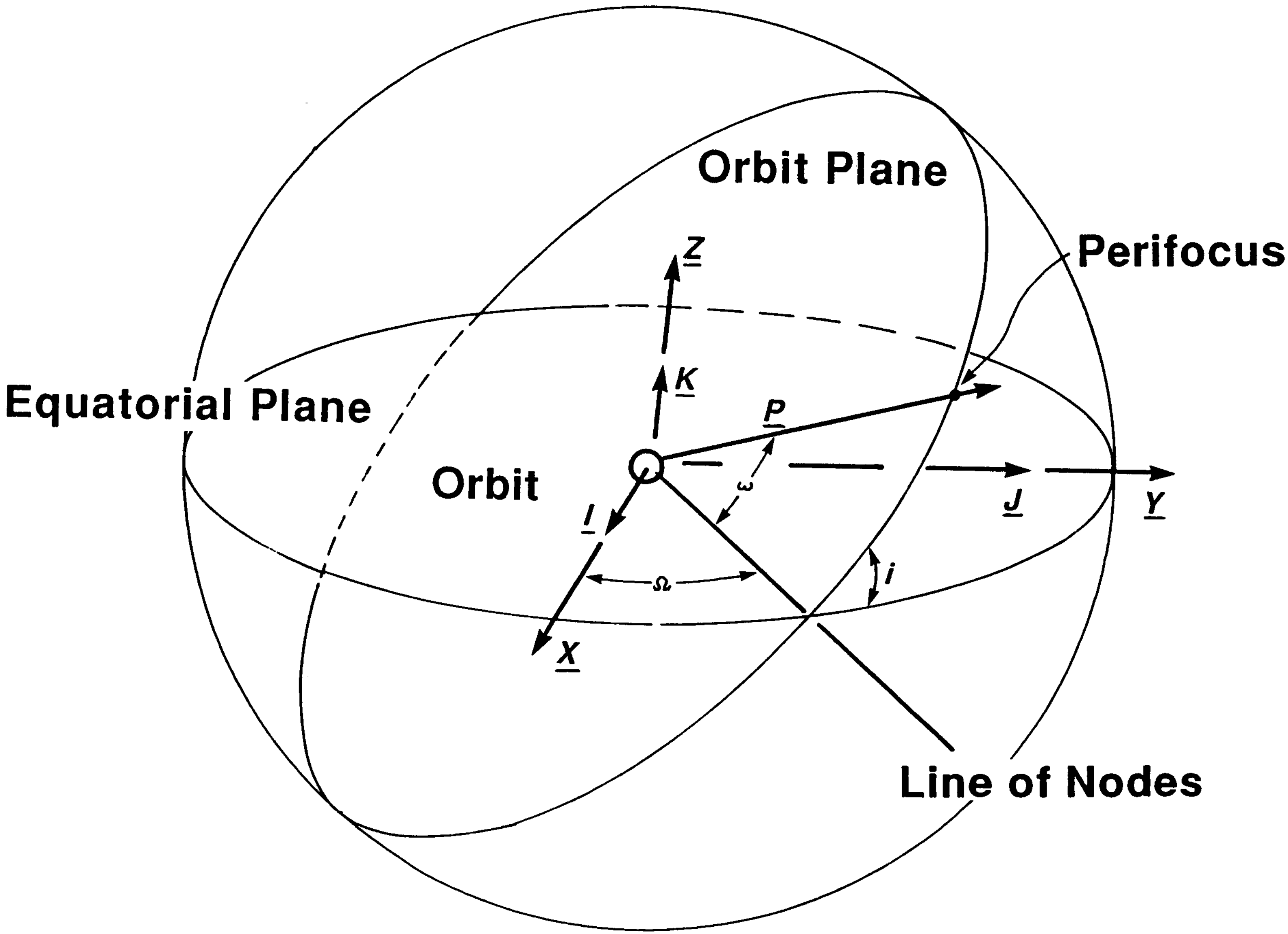


After integrating and obtaining the new state vector, the main program checks to see if the MECO time has been exceeded. If so, then the main program exits the integration loop and begins final orbit calculations. Otherwise, it loops back to the point just after Variable Initialization and performs another integration.

At specific time increments the program outputs important information about the flight. During Variable Initialization, the output time increment is set to five (5) seconds. The information presented to the screen during this intermediate output phase includes the time, altitude, range, velocity, flight path angle, heading, acceleration, thrust, and weight. In addition to this information, data such as rate of change of flight path angle, longitude, and latitude are output to a file called "LOUTPUT.PRN" ("LOUTPUT.DAT" in FORTRAN versions).

The final/initial orbit is evaluated in the Orbit subroutine. This subroutine calculates the apocynthion, pericynthion, inclination, longitude of the ascending node, argument of pericynthion, and the eccentricity of the orbit entered. These terms are known as orbital elements and are shown graphicly in Figure 2. They are printed to the screen and to the output file.

Figure 2: Orbital Elements



3.0 DATA ENTRY SUBROUTINE

The Data Entry subroutine is the section of the program that asks the user for the information required to run the simulation. In BASIC versions, the user is first prompted for the letter designation of the storage drives used for input and output.

> Drive for Input data files ---------------------
> Drive for Output data files ---------------------
> Choose 'F' for File Entry or 'M' for Manual Entry.
> Is this to be an Ascent or a Descent simulation ?

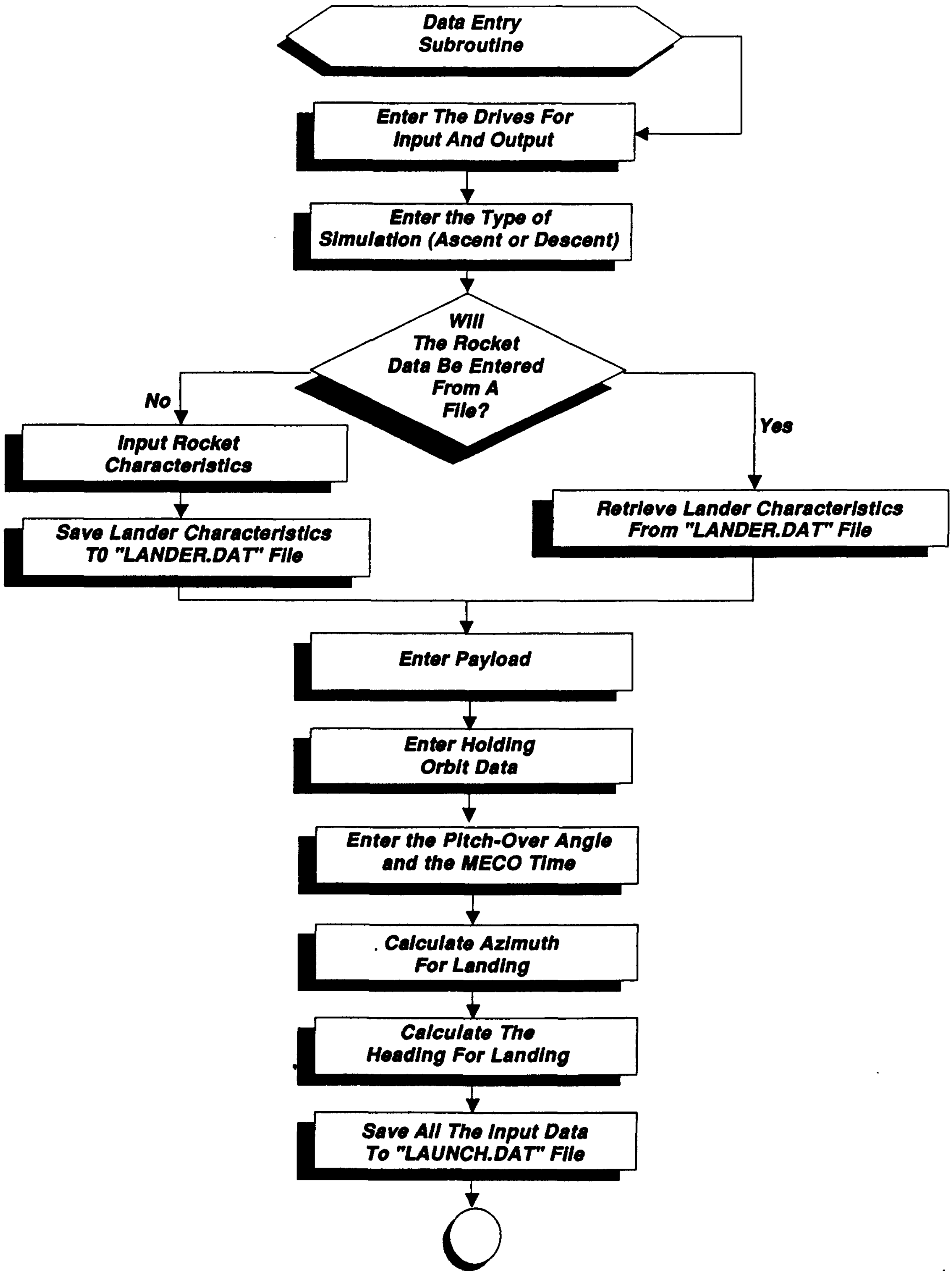
Different drives may be assigned to perform the input and output functions. Storage drives are only important for the versions of this program that are used by personal computers (PCs). On PCs, it is common to save input and output data on disks when it is being held for archive purposes. For working files, disk access is a slow process; storage and retrieval of data from the Random Access Memory (RAM) is much faster. Therefore it is common to transfer archived data to the RAM drive, and then assign the RAM drive to handle all input and output operations.

BASIC versions of LANDER also allow the lander characteristics to be provided via an input file. If the user selects this option, he or she should type an "F" or "f" when prompted with, "Choose 'F' for File Entry or 'M' for Manual Entry." Any answer other than "F" or "M" will result in the question being restated.

If an ascent simulation is desired, then the user should type an "A" when asked for the type of simulation. A descent simulation can be performed by typing a "D". Entry of any set of characters not beginning with an "A" or a "D" will result in the question being reprompted.

The longitude and latitude of the landing site must be specified on the next input screen (See Below). Longitude meridians are measured in degrees east of the Prime Meridian which passes through the Earth-Moon line on the Earth side of the Moon. Values between 0° and 360° East longitude can be used. The latitude is measured north from the lunar equator. Southern latitudes are indicated as a negative. Latitudes between 90° and -90° North latitude are permissible.

Figure 3: Data Entry Flow Chart



Lunar Landing Site
Landing Site Latitude (-90 to +90)
Landing Site Longitude (0 to 360)

***** Vehicle Configuration *****

Payload Weight <lb>

Inert Weight <lb>	Propellant Weight <lb>
Thrust <lbf>	Specific Impulse <s>
Hover Time <s>	

If the user chose to input the lander characteristics manually, then the inert weight, fuel weight, maximum thrust, specific impulse, and hover time of the lander must be provided. The program will then save the information on the output drive in a file called "LANDER.DAT". If the file entry method is adopted, then the program will search on the input drive for the "LANDER.DAT" file, and read the data in that file. Keep in mind that this option is only available for BASIC versions of LANDER.

The inert weight is the weight of the structure and equipment necessary for spacecraft operation. (All weights are Earth weights; the force measured by a scale on the surface of the Earth.) The propellant weight is supplied for ascent purposes. For ascent the weight of the propellant must be known in advance. It is added to the inert weight and the payload weight to obtain the spacecraft weight prior to lift-off. During descent simulations, the propellant weight is calculated and does not necessarily need to be input. The maximum thrust must be included since the thrust profile (discussed in section 6.2) is normalized to the maximum thrust. A constant propellant specific impulse is assumed throughout the flight, and the hover time can be of any length requested by the user.

The user is then queried for a payload weight. The payload is a constant mass element which is not an integral portion of the lander structure (i.e. not part of the inert weight).

The next input screen appears as follows:

Time to Main Engine Cut-off (MECO) ? _____ <s>
Holding Orbit (_______ <nm> X _______ <nm>)

The spacecraft will perform a vertical rise (Flight Path Angle {Gamma} = 90 deg.) for the first few seconds of flight. At a relative velocity of 30 ft/s a pitch-over maneuver is executed; and the vehicle will momentarily thrust along a flight path defined by the user (Good Value = 70°)

Flight path angle at pitch-over ? ___

Holding orbit inclination ? (0° to 360°) ____
Do you wish to see the trajectory of each iteration ? __

The user must provide an initial estimate to the simulation run time. The simulation runs until Main Engine Cut-off (MECO). 300 seconds is typical for an ascent simulation, while 450 to 500 seconds are good values for descent simulations. The holding orbit is the orbit from which the lander will begin its descent or to which the ascent spacecraft will inject after launching from the Moon.

The program requests that the user supply an initial value for the pitch-over flight path angle. If the flight path angle at the end of the simulation (MECO) is greater than zero (0), then the pitch-over angle is too high and the simulation is rerun with a lower pitch-over angle. The reverse is true if the flight path angle at MECO is less than zero (0). The process is iterative, and it requires several attempts to obtain the proper flight path angle at MECO. If the flight path angle at MECO is zero (0), then the final orbit is analyzed. If the resulting orbit is too high, then the simulation is terminated sooner (MECO time is reduced). Using the shorter simulation time the final flight path angle may not be zero (0), and must, therefore, be reiterated. If the resulting orbit is too low, then simulation is terminated later. Again, the process is iterative, and several modifications of the MECO time are necessary to obtain a solution.

The holding orbit inclination must be greater than the latitude of the landing site. If the landing site is at 45 degrees North or South latitude, then the true orbit inclination must be at least 45 degrees. From a mathematical point of view the orbit can never have a true inclination of more than 90 degrees. However, the latitude of the launch site and the true inclination of the orbit are not sufficient to define the direction from which the lander will make its approach. As Figure 4 demonstrates a lander attempting to land at site "A" from an orbit of true inclination "i" can be approaching from four different directions.

Posigrade orbits, those traveling in the direction of planetary rotation -- left to right, and retrograde orbits, traveling opposite the planetary rotation -- right to left, can approach a specified landing site from either the North or South. In order to show from which direction the lander is approaching the landing site, or the ascent spacecraft is heading from the launch site, this program allows the user to input an inclination that may be greater than 90°. If the input inclination is less than 90° (i.e. the input inclination equals the true inclination) then the spacecraft is flying from South to North in a posigrade orbit (Case I of figure 4). If the spacecraft is flying from South to North in a retrograde orbit, then the user supplies an inclination (pseudo-inclination) that is greater than 90° but less than 180° (See case II of figure 4). For a true inclination of "i" (between 0° and 90°), Table 1 shows how to calculate the pseudo-inclination which should be input to the program at the "inclination" prompt. When the flight is from North to South in a retrograde orbit, then the pseudo-inclination is between 180° and 270° (Case III -- figure 4). Finally for North to South flights in a posigrade orbit, the user should input a pseudo-inclination between 270° and 360° (Case IV -- figure 4). If the user inputs a pseudo-inclination which is less than 0° or greater than 360°, then the program will reprompt for the inclination.

Table 1: Calculation of the Pseudo-inclination

Case	Orbit Type	Direction of Flight	Pseudo-inclination Equals
I	Posigrade	From S. to N.	i
II	Retrograde	From S. to N.	180° - i
III	Retrograde	From N. to S.	180° + i
IV	Posigrade	From N. to S.	360° - i

The approach azimuth is calculated from the pseudo-inclination and the latitude of the landing/-launch site through the use of right-spherical triangles (Figure 5).

The approach azimuth is an angle between -90° and 90°. Table 2 is used to relate the approach azimuth to the approach heading. The heading is measured from the North clockwise, and has a value between 0° and 360°. This is shown graphically in Figure 6.

Figure 4: Approach Paths for Landing

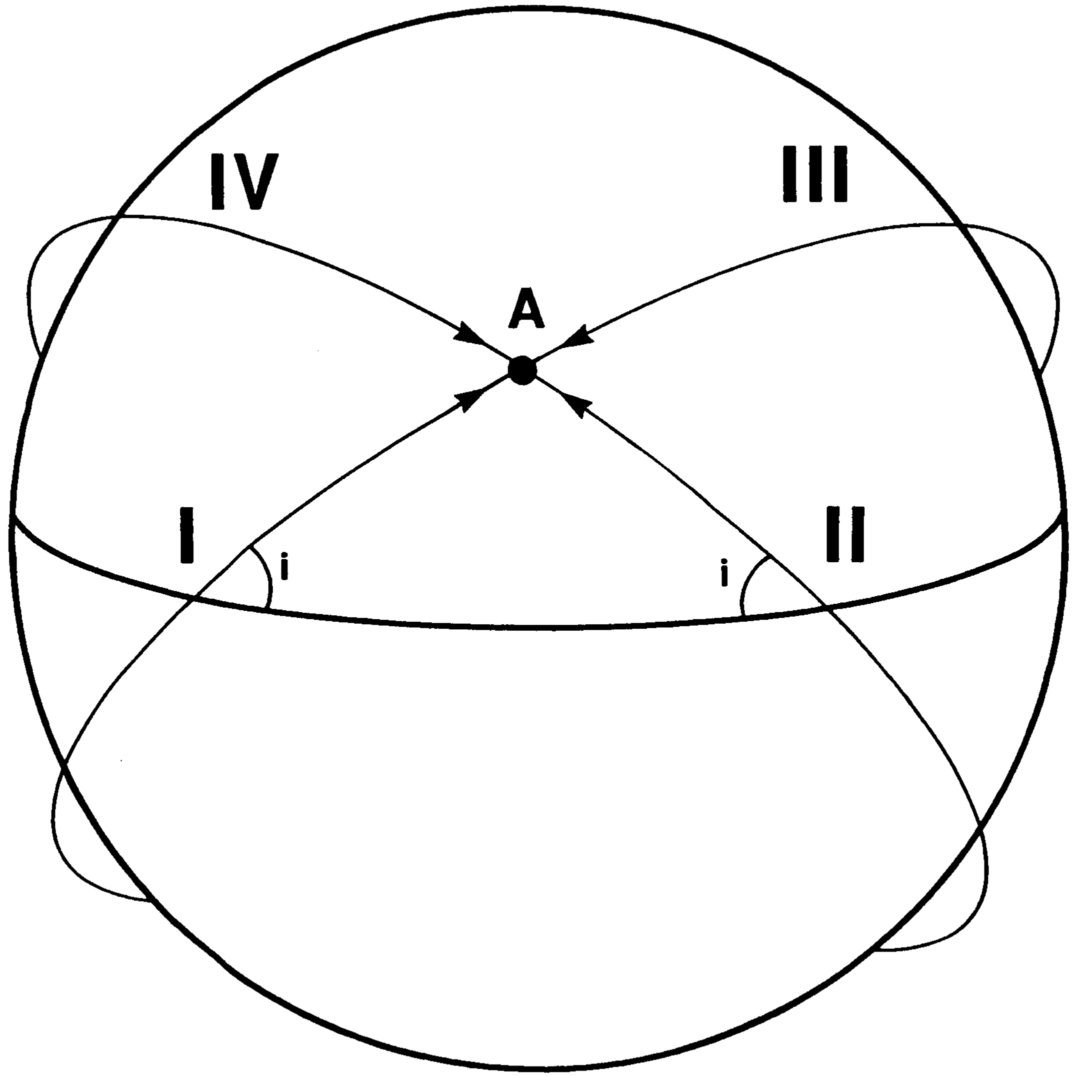
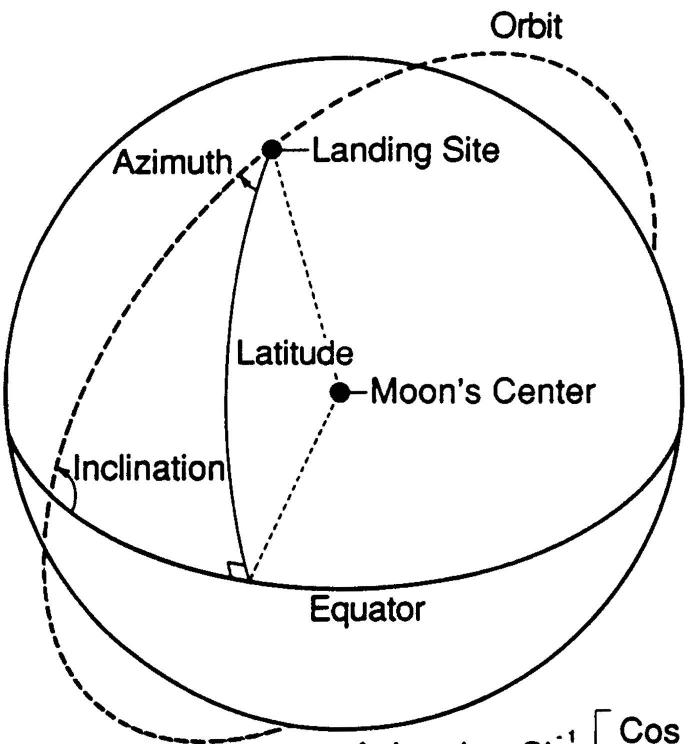


Table 2: Heading and Azimuth Relationship to Pseudo-inclination

Pseudo-Inclination	Approach Azimuth (AZH)	Approach Heading
0° to 90°	Positive	AZH
90° to 180°	Negative	360° + AZH
180° to 360°	Pos. or Neg.	180° + AZH

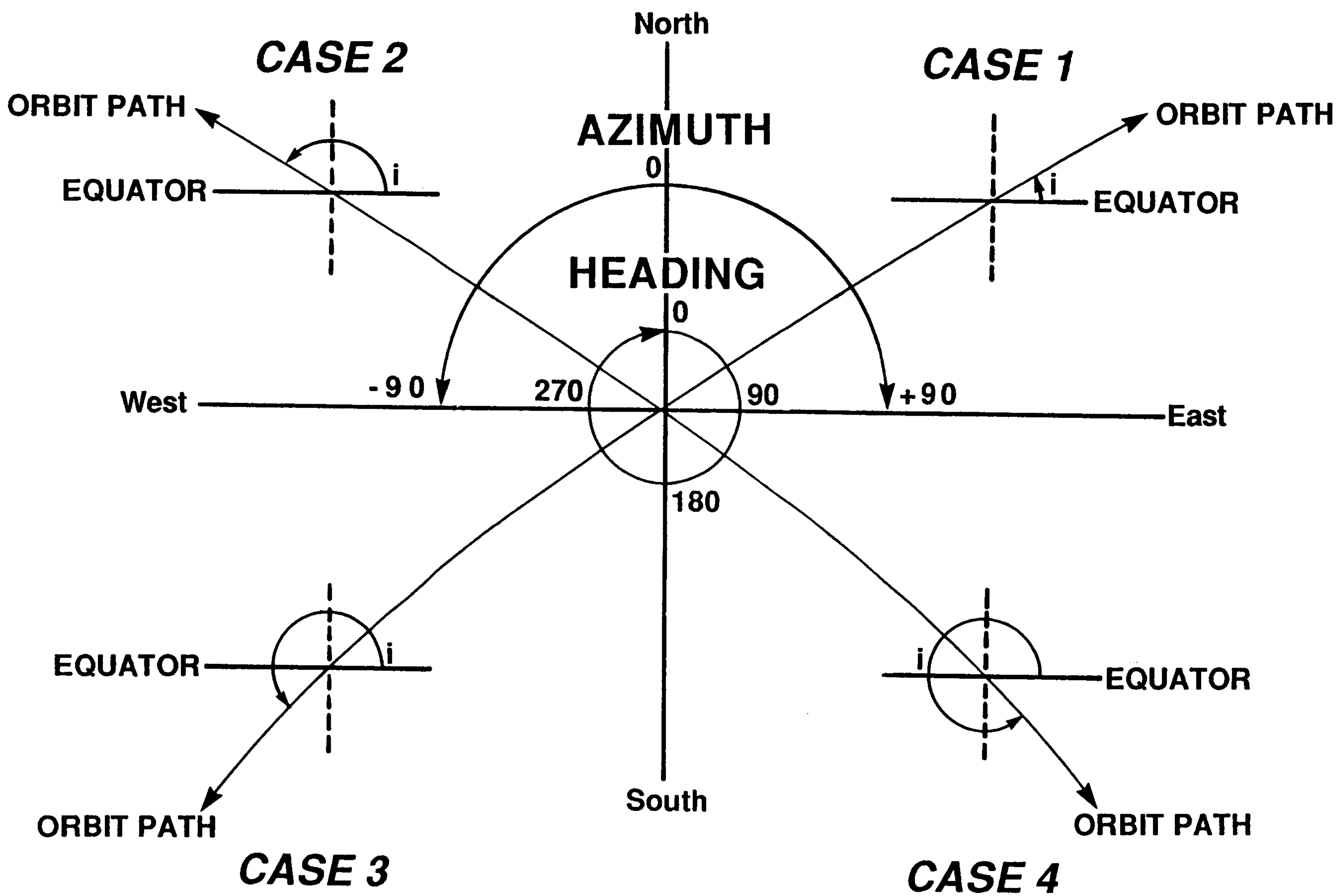
During each iteration of the trajectory, the program calculates numerous variables. The data stored in these variables can be sent to the screen if desired. However, if the program is running properly, then the extra output is unnecessary and time consuming. The trajectory data output can be turned off by answering "No" when ask, "Do you wish to see the trajectory of each iteration?"

Figure 5: The Aproach Azimuth



$$\text{Azimuth} = \sin^{-1}\left[\frac{\cos(\text{inclination})}{\cos(\text{latitude})}\right]$$

Figure 6: Heading, Azimuth, and Inclination



4.0 VARIABLE INITIALIZATION SUBROUTINE

A trajectory may be run numerous times during a simulation. It is important that each run start with the same initial conditions, and that stored data is not randomly retrieved during the simulation. During Variable Initialization all variables that are to be used during the simulation are set to their initial values. At the end of Variable Initialization the state vector, describing the initial conditions under which the vehicle is operating, is formulated.

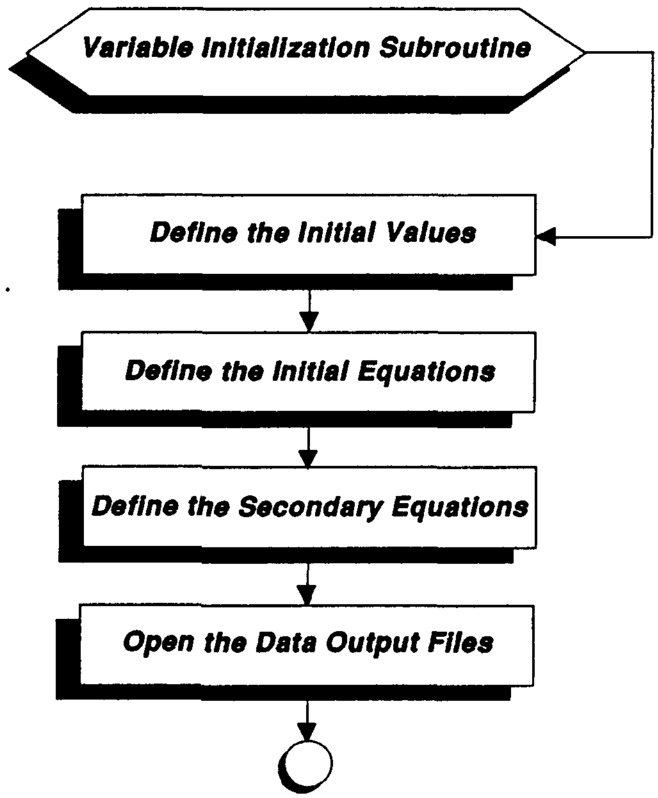
The state vector describes characteristics of the vehicle which are not constant. These characteristics are referred to as parameters. Examples of "state" parameters include: the vehicle's position, velocity, and mass. The payload and inert weight are not considered "state" parameters because these vehicle characteristics are constant.

There are seven parameters which define the present "state" of the vehicle. The position is described by three parameters: the distance, the longitude angle, and the latitude angle. The mass of the vehicle represents the fourth state parameter. The vehicle's velocity is also described with three parameters: the radial range rate, the angular rate of longitude, and the angular rate of latitude.

All seven parameters form a vector or an array which, when integrated, creates a new state vector. The new state vector describes the conditions of the vehicle in the future; at the end of the time step. The ability to integrate the state vector is what makes it possible to determine the new position, velocity, and mass of the vehicle at the future time.

At the end of the Variable Initialization subroutine, a data storage file, called "LOUTPUT.PRN", is opened on the output drive. The file has a "PRN" extension so that it can be recognized by LOTUS (A spreadsheet programming language) as an input/output data file. In FORTRAN versions, this file is called "LOUTPUT.DAT".

Figure 7:
Variable Initialization Flow Chart



5.0 INTEGRATION SUBROUTINE

LANDER makes use of a Runge-Kutta fourth order integration routine. The derivation of this integrator is discussed in the second edition of Curtis F. Gerald's Applied Numerical Analysis, on page 259. A summary of this discussion and how it applies to the program is necessary in order to clarify the coding process.

The primary equation is obtained by substituting "t" for "X", and then "X" for "Y" in the equation presented by Gerald. In this equation the "K" values are estimates of the X. The weighted average of these estimates is the increment to X_{n+1} from X_n.

1) $X_{n+1} = X_n + (K1 + 2*K2 + 2*K3 + K4) / 6$

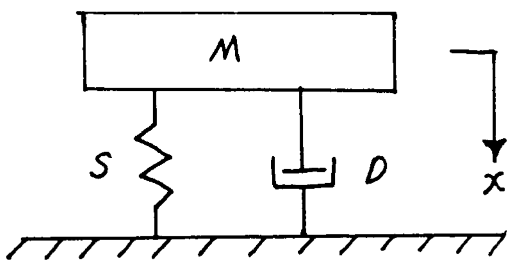
Where:

$$K1 = dt * f(X_n \quad , t_n \quad)$$
$$K2 = dt * f(X_n + K1/2, t_n + dt/2)$$
$$K3 = dt * f(X_n + K2/2, t_n + dt/2)$$
$$K4 = dt * f(X_n + K3 \quad , t_n + dt \quad)$$

dt = Time step
t = Independent time variable
X = Time dependent state variable

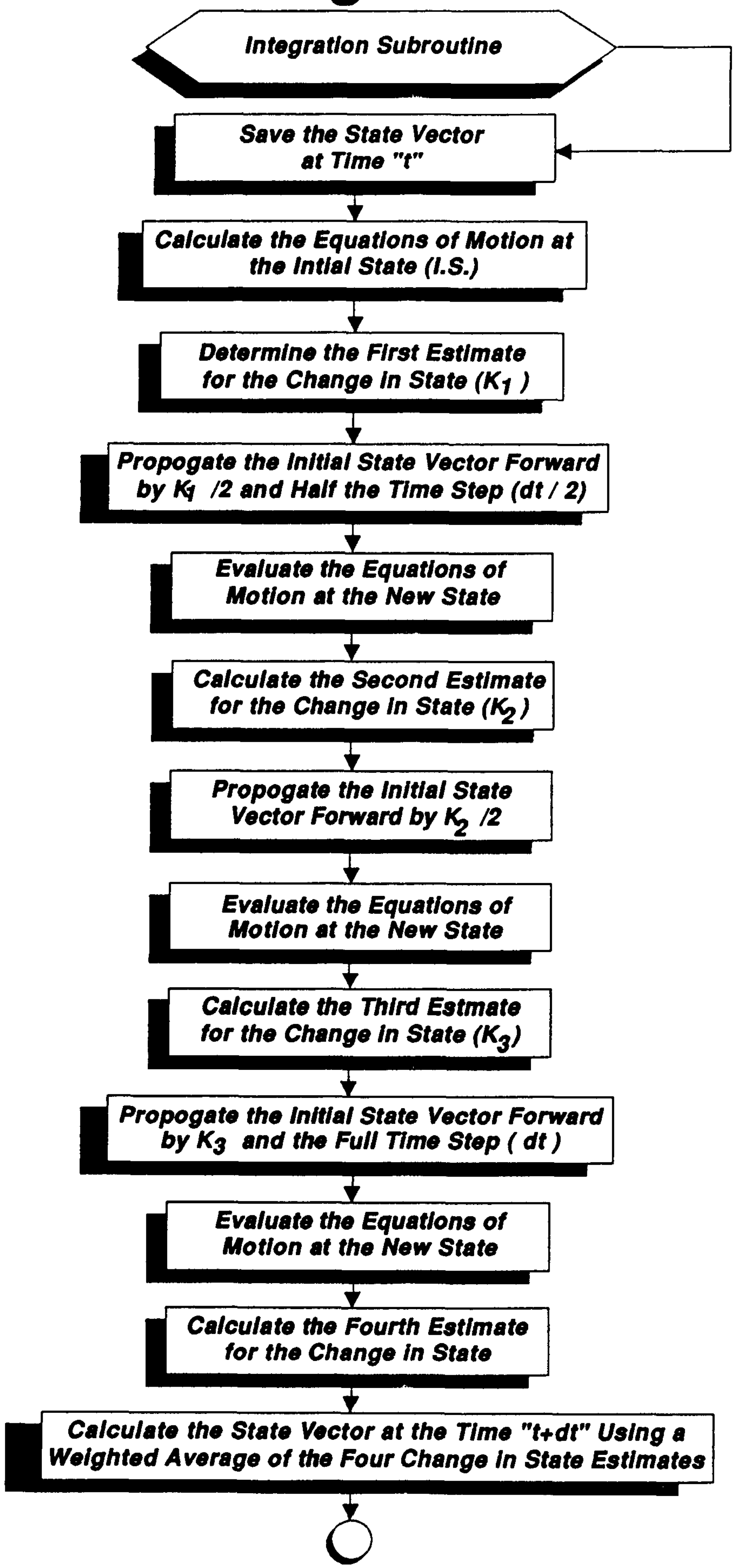
The function "f" is known as the Equation of Motion (EOM); and "X_n" is the state vector at time "t".

Example: In the spring (S), mass (M), damper (D) system shown,



the Equation of Motion is: $$M * \frac{d^2x}{dt^2} + D * \frac{dx}{dt} + S*x = 0$$

Figure 8: Integration Flow Chart



The equation of motion is usually written in following form.

$$f = \frac{d^2x}{dt^2} = -\frac{D}{M} * \frac{dx}{dt} - \frac{S}{M}*x$$

K1 is calculated by evaluating the EOM at state "X_n" and taking the product with the time step. Subsequently, K2 is determined after evaluating the EOM at the new state of "X_n + K1/2", and the new time "t_n + dt/2". K3 and K4 are similarly obtained after two more evaluations of the EOM. Once the "K" values have been determined, "X_{n+1}", the state vector at time "t + dt", is calculated using Equation 1.

6.0 EQUATIONS OF MOTION

Numerous calculations are necessary in order to set up the equations of motion. These preliminary calculations must be complete before integration can proceed.

The overall control parameter is the time variable. The time is what the program uses to determine when to stop the simulation. If the simulation time has not exceeded the user supplied stop time, then the main program continues to increment the time and loop back to the Integration subroutine. When the simulation time does exceed the stop time, the main program transfers control to the Orbit calculation subroutine, and then proceeds to the final output sequence.

The velocity is determined from the state vector in spherical coordinates ($R, \theta, \phi, M, \dot{R}, \dot{\theta}, \dot{\phi}$. "R" is the radial distance from the center of the Moon. "θ" is the angle of longitude, measured East from the Prime Meridian. "ϕ" is the angle of latitude, measured North from the Equator. "M" is the mass of the lander. "$\dot{R}$" is the radial range rate outward from the Moon's center. "$\dot{\theta}$" is the angular rate of change of longitude. And "$\dot{\phi}$" is the angular rate in latitude. The inertial velocity is calculated with the following equations:

INERTIAL VELOCITY COMPONENTS

$$V_r = \text{Radial Velocity} = \dot{R}$$

2) $$V_t = \text{Longitudinal Velocity} = R * \dot{\theta} * \cos(\phi)$$

$$V_p = \text{Lateral Velocity} = R * \dot{\phi}$$

Inertial speed is the root sum square of the three velocity components shown above. The local velocity can be determined by reducing the longitudinal velocity component by the rotation rate of the moon.

LOCAL VELOCITY COMPONENTS

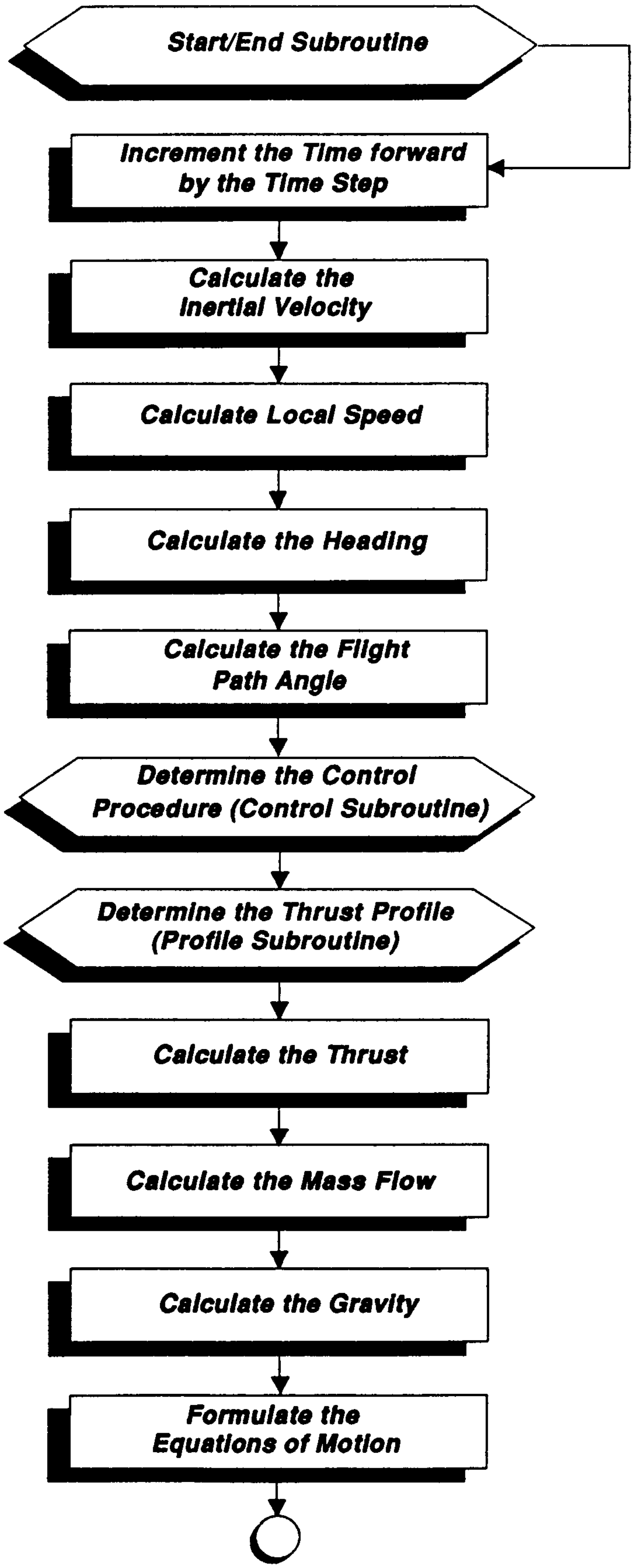
$$Vl_r = V_r$$

3) $$Vl_t = V_t - R * \text{Omega} * \cos(\phi)$$

$$Vl_p = V_p$$

Where: Omega = Rotation rate of the Moon = 2.26622×10^{-6} <rad/s>

Figure 9: Equations Of Motion Flow



The altitude above the surface of the Moon is calculated. Orientation of the thrust vector (GAMT) is determined in the Control subroutine. The level of thrust (PRF) is calculated in the Profile subroutine. The thrust (T1), the propellant mass flow (MDOT), the local acceleration of gravity (G), the heading (HEAD), weight (WEIGHT), and thrust to weight (TTOW) are also calculated in this section.

The program uses spherical coordinates to evaluate the motion of the lander. In spherical coordinates, "r" is the radial distance, "θ" is the longitude angle from the inertial X axis, and "ϕ" is the latitude angle measured from the equator.

In spherical coordinates, the equations of motion for a spacecraft of mass (m) under the influence of thrust (T) and gravity (g) are given in Equation 4.

$$\ddot{r} = r\dot{\theta}^2[\mathrm{Cos}(\phi)]^2 + r\dot{\phi} + \frac{T}{m}*\mathrm{Sin}(\gamma) - g$$

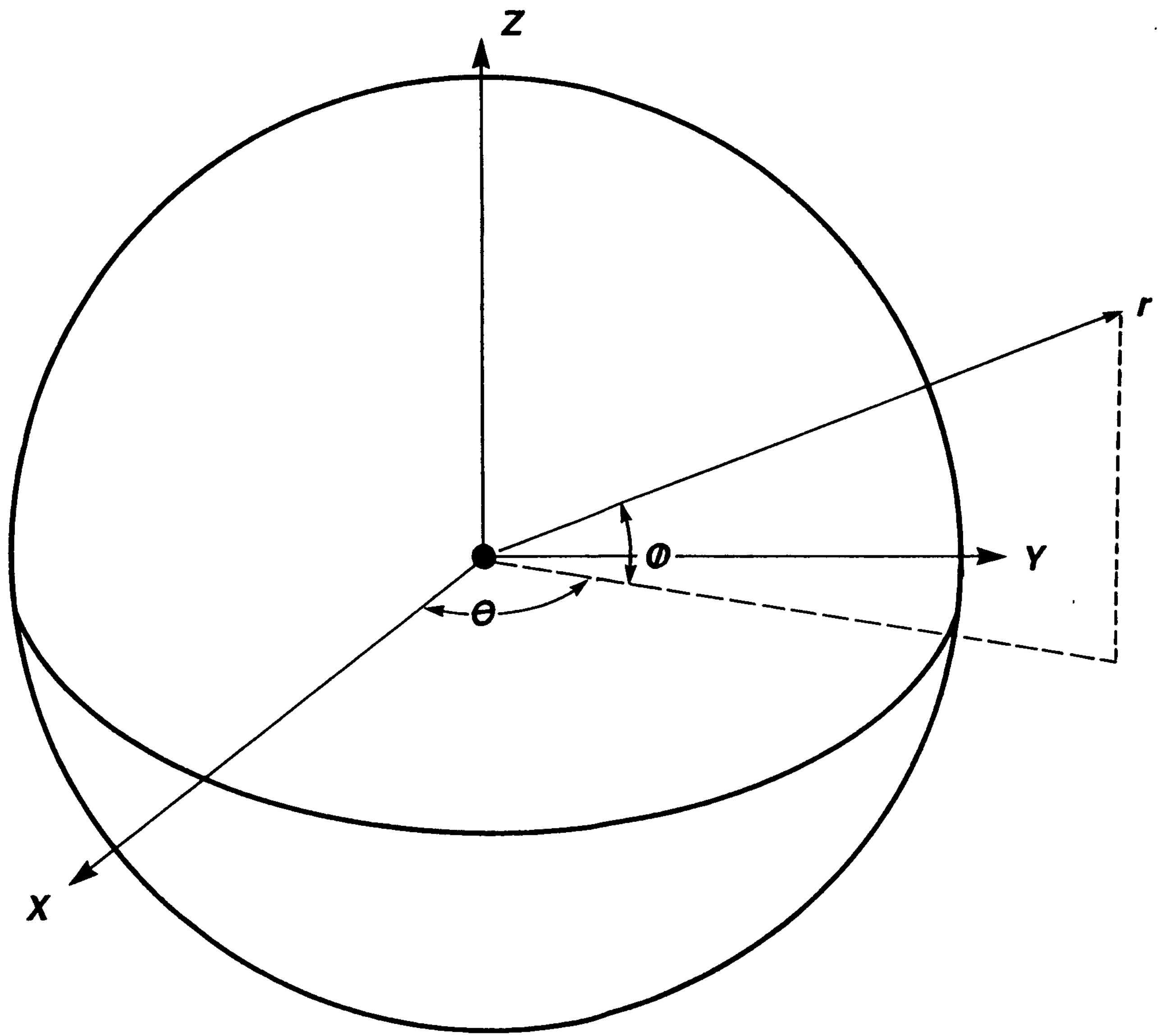
$$4)\quad \ddot{\theta} = \frac{2\dot{\theta}\dot{\phi}*\mathrm{Sin}(\phi)}{\mathrm{Cos}(\phi)} - \frac{2\dot{r}\dot{\theta}}{r} + \frac{T*\mathrm{Cos}(\gamma)*\mathrm{Sin}(h)}{mr*\mathrm{Cos}(\phi)}$$

$$\ddot{\phi} = \frac{T*\mathrm{Cos}(\gamma)*\mathrm{Cos}(h)}{mr} - \dot{\theta}^2*\mathrm{Cos}(\phi)\mathrm{Sin}(\phi) - \frac{2\dot{r}\dot{\phi}}{r}$$

Where: γ = Flight Path Angle
h = Heading

The flight path angle is measured up from the local horizon, and the heading is measured clockwise from North.

Figure 10: Spherical Coordinates



6.1 CONTROL PROCEDURES

The Control subroutine provides the thrust orientation for the lander throughout the descent. The thrust vector is controlled through a pitch angle (GAMT). The thrust pitch angle can vary from 0° (tangential to the lunar surface) to 90° (normal to the surface).

The lander begins its descent from orbit using a gravity turn trajectory. As it slows the flight path angle gradually increases from 0°. Ten (10) seconds before the velocity reaches 30 ft/s, the lander initiates the pitch-over maneuver which is designed to reduce the horizontal velocity to zero. The thrust pitch angle is reoriented to the pitch-over angle (GAMP) during the next five (5) seconds. Then it orients to 90° (vertical) during the following five (5) seconds.

At the end of the pitch-over maneuver the lander is descending at 30 ft/s and has no horizontal velocity. The lander continues to decelerate until it is descending at 1.6 ft/s, basicly hovering. A 1.6 ft descent/hover is then maintained until touchdown. The altitude at which the lander reaches 1.6 ft/s descent velocity is dependent upon the amount of time that the user wishes the vehicle to hover.

During ascent, the spacecraft launches at full thrust vertically until it reaches 30 ft/s local velocity. It performs a ten second pitch-over maneuver, and flies a gravity turn to orbit.

Figure 11: Thrust Pitch Angle

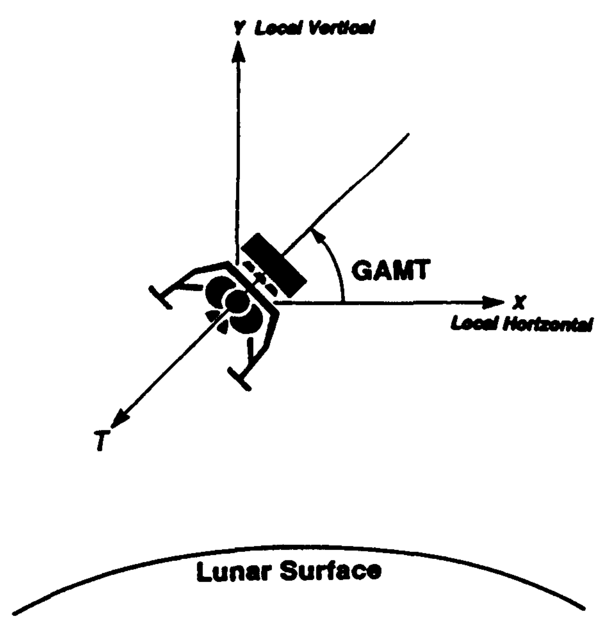
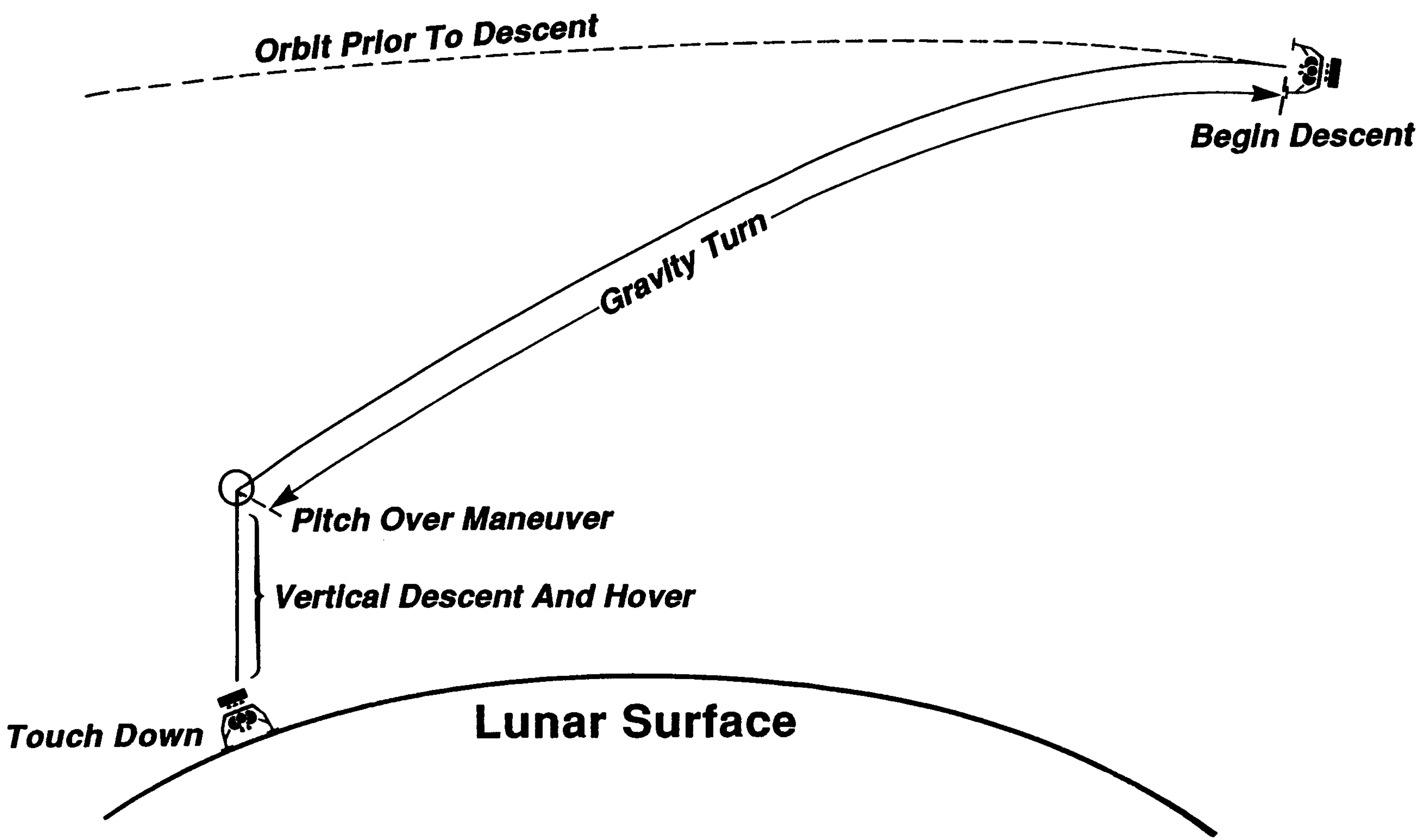


Figure 12: Control Model



6.2 THRUST PROFILE

The thrust profile subroutine is accessed in the preliminary calculations of the Equations of Motion subroutine. This subroutine returns the level of thrust (PRF) as a percentage of the maximum thrust. The thrust level is dependent on time and local weight. From initiation of the descent to 35 seconds prior to hover the thrust level is set to maximum thrust. During the next 35 seconds, the thrust is linearly reduced to a level that is equal to the local weight of the lander. During this 35 seconds, the vertical descent velocity is reduced to 1.6 ft/s, and the horizontal velocity is nulled during pitch-over.

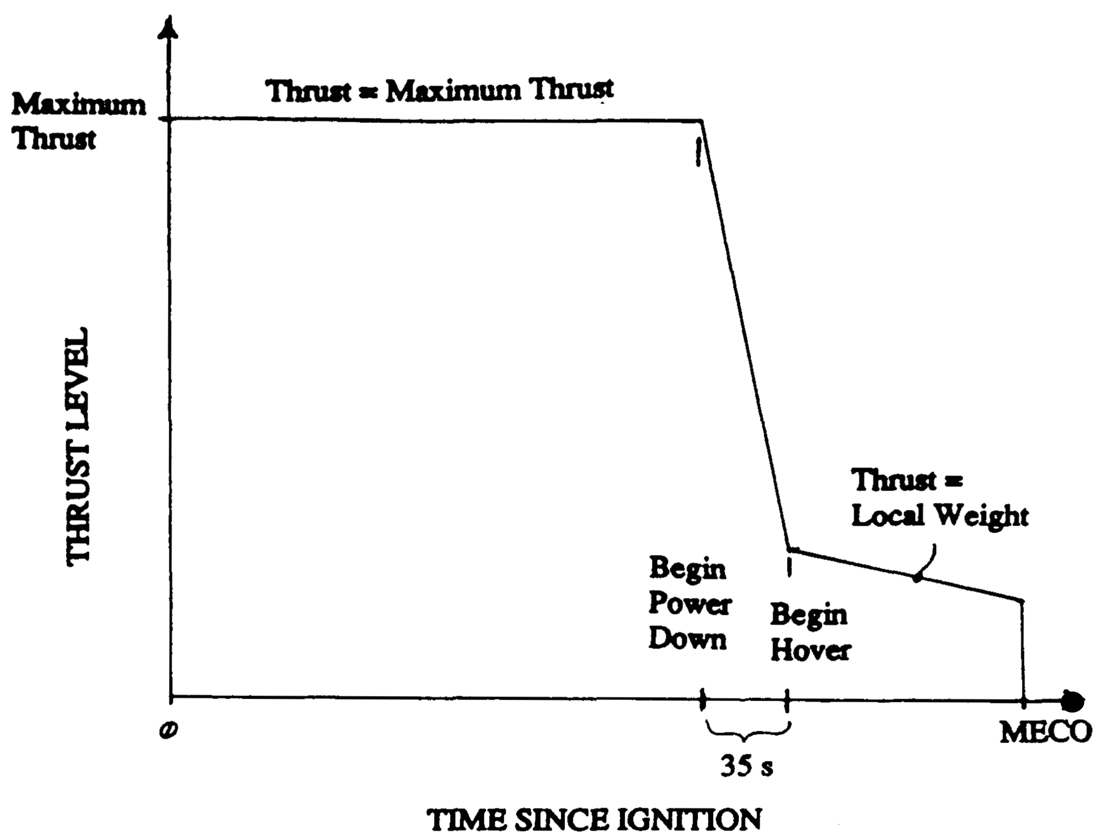


Figure 13: Lander Thrust Profile

The thrust profile for an ascent from the surface is a constant, and is held at maximum thrust. Thrust profile modifications can be accomplished by rewriting the Thrust Profile subroutine.

7.0 OUTPUT SUBROUTINE

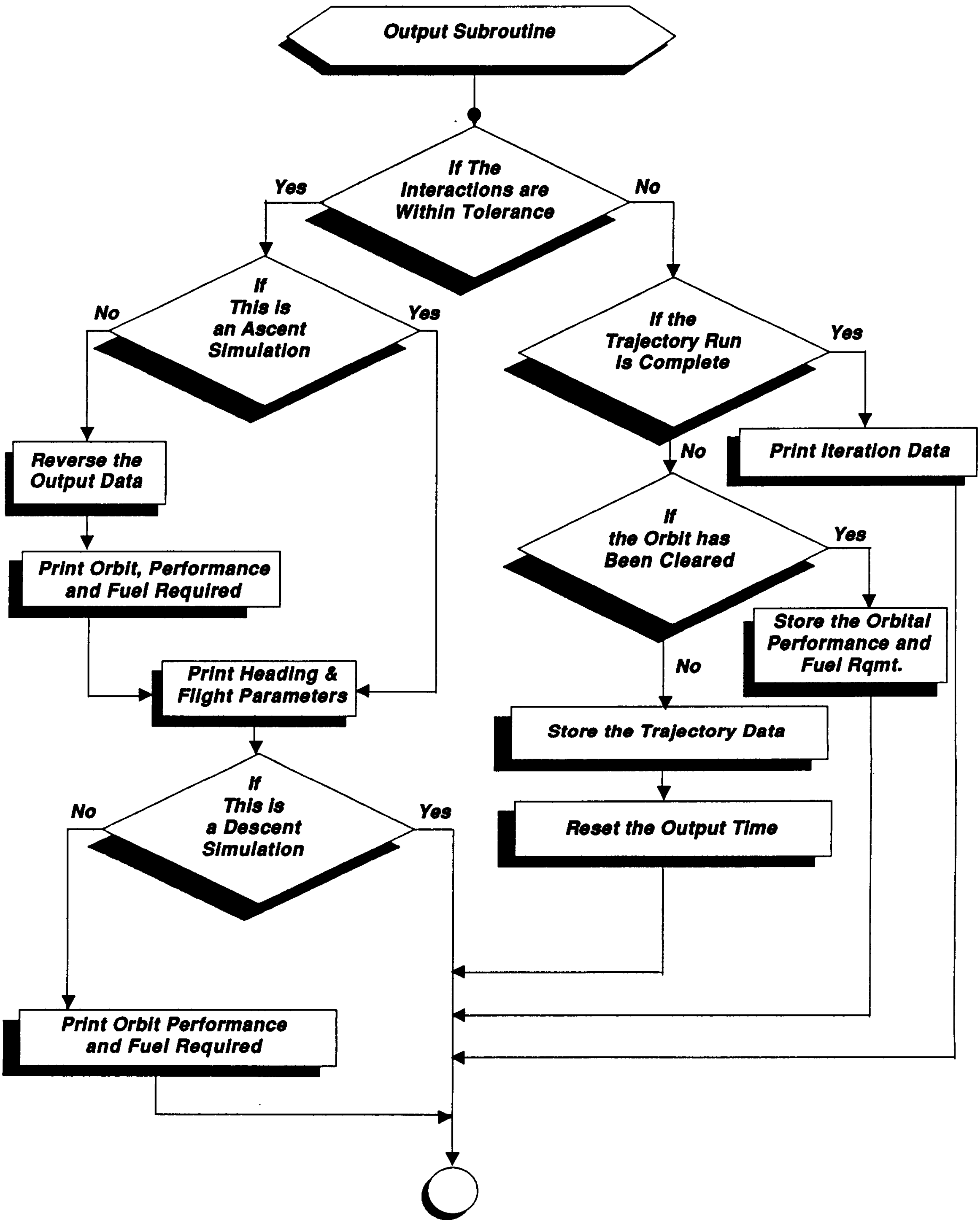
The output subroutine is the portion of the program that controls when, where, and what information is to be presented to the screen and data storage files. The output data file is called LOUTPUT.PRN (LOUTPUT.DAT in FORTRAN version). This file has a ".PRN" extension in the BASIC version which allows it to be recognized as a data file by LOTUS. LOTUS is a spreadsheet program used to create graphical output from mass data. Other graphics programs may be used as long as they can read ASCII sequential data files.

Every 5 seconds of simulation time the output subroutine prints to the screen the time, altitude, range, velocity, flight path angle (Gamma), heading, thrust/weight, thrust, and the weight. Examples of typical ascent and descent screen output are provided in Appendix C. In addition to this data, longitude, latitude, angle of attack, rate of change of angle of attack, and the rate of change of flight path angle are saved to the LOUTPUT.*** file. The velocity and flight path angle are presented in local coordinates. When using local coordinates, the velocity and flight path angle are always given with respect to the launch site.

When the program is finished the output subroutine displays and saves the orbital parameters and the performance delta velocity. The orbital parameters consist of the apocynthion, pericynthion, inclination, longitude of the ascending node, argument of pericynthion, and the eccentricity (refer to Figure 2). The performance delta velocity (ΔV) is the ideal velocity change that could be made with the fuel used if there are no gravity losses.

If the output interval needs to be changed, then it can be changed manually in the Initialization subroutine. The variable to be changed is called OUTINT. If the output interval is less than the integration step size (DT), then output will occur during each integration step.

Figure 14: Output Flow Chart



8.0 ORBIT SUBROUTINE

The orbit subroutine calculates the orbital elements of the orbit from which the lander is to descend. The orbital elements of interest are the apocynthion altitude, the pericynthion altitude, the inclination (i), the longitude of the ascending node (Ω), the argument of pericynthion (w), and the eccentricity (e). Figure 2 is useful for visualizing these elements. The pericynthion altitude is the altitude of the spacecraft when it is at the perifocus of the orbit. Apocynthion altitude is the altitude when the spacecraft is opposite the perifocus. The eccentricity of the orbit is a measure of its ellipticity.

The orbital elements are calculated from the position and velocity vectors. The velocity vector must be in radial coordinates, and the position vector needs to be in rectangular inertial coordinates.

Radial coordinates are defined such that the X axis is aligned with the radial position vector from the center of the planet, the Y axis is parallel to the equatorial plane, and the Z axis is normal to the X-Y plane (Figure 16).

Since the velocity vector is normally in inertial coordinates (V_i), it must be converted to radial coordinates (V_r). This can be accomplished with Equation 5. The conversion to radial coordinates "r" from inertial coordinates "i" is achieved through vector multiplication of successive orthogonal rotation matrices for the longitude rotation (θ) and the latitude rotation (ϕ).

5) $$\underline{V_r} = [M(\phi)]\ [M(\theta)]\ \underline{V_i}$$

Where: $$[M(\phi)] = \begin{vmatrix} \cos(\phi) & 0 & -\sin(\phi) \\ 0 & 1 & 0 \\ \sin(\phi) & 0 & \cos(\phi) \end{vmatrix}$$

$$[M(\theta)] = \begin{vmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

$$[C(\theta,\phi)] = [M(\phi)]\ [M(\theta)]$$

$$= \begin{vmatrix} \cos(\phi)\cos(\theta) & \cos(\phi)\sin(\theta) & -\sin(\phi) \\ -\sin(\theta) & \cos(\theta) & 0 \\ \sin(\phi)\cos(\theta) & \sin(\phi)\sin(\theta) & \cos(\phi) \end{vmatrix}$$

Figure 15: Orbit Flow Chart

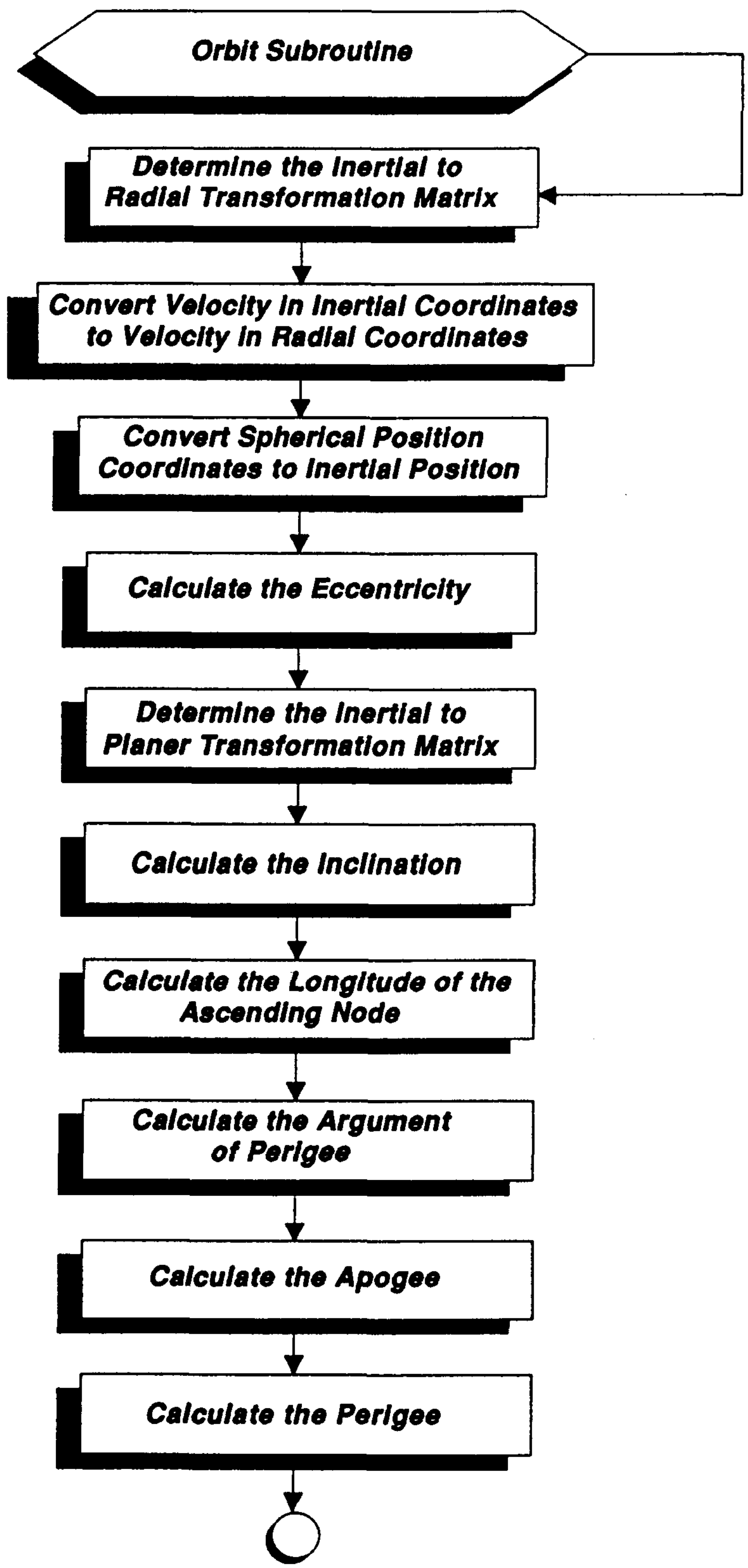
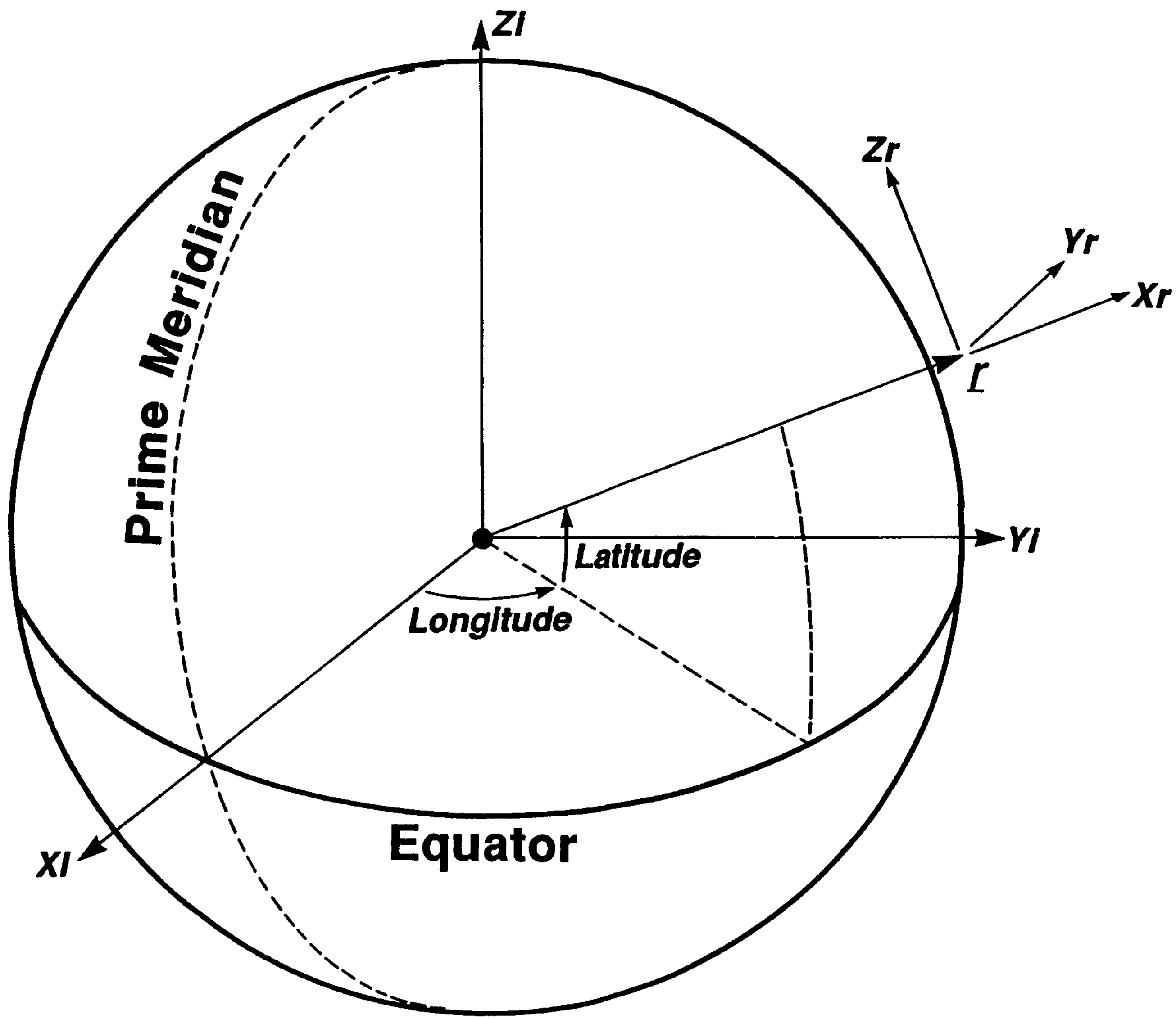


Figure 16: Radial Coordinates



As mentioned earlier, the position vector must be in rectangular inertial coordinates, but it is given in spherical coordinates where "R" is the radial distance from the center of the planet, "θ" is the angle of longitude from the Prime Meridian (X axis), and "ϕ" is the angle of latitude from the equatorial plane. This conversion is shown in Equation 6.

$$
\begin{aligned}
X &= R * \cos(\phi) * \cos(\theta) \\
Y &= R * \cos(\phi) * \sin(\theta) \qquad (6) \\
Z &= R * \sin(\phi)
\end{aligned}
$$

Once the velocity and position vectors are in the proper coordinates, the orbital elements are calculated. A complete discussion of the calculation of orbital elements is beyond the scope of this report; but for those interested in the subject, a good treatment can be found in Chapter 17 "Satellite Photogrammetry" written by John L. Junkins from the Manual of Photogrammetry, 4th ed., American Society of Photogrammetry, Falls Church, Va., 1980.

APPENDIX A: VARIABLE DEFINITIONS

Table A1: Variable Arrays

C(3,3)	-	Coordinate System Transformation Matrix
K1(15)	-	1^{st} Estimate for the Change of State
K2(15)	-	2^{nd} Estimate for the Change of State
K3(15)	-	3^{rd} Estimate for the Change of State
K4(15)	-	4^{th} Estimate for the Change of State
M(4)	-	Mass Array
PSN(3)	-	Position Array
RKX(15)	-	Runge-Kutta State Vector
RKDX(15)	-	Runge-Kutta State Derivative
SDAT(5)	-	Stage Data Array
TRAJDAT(100,20)	-	Storage Array for Trajectory Data
VEL(3)	-	Relative Velocity Array
VSP(3)	-	Inertial Velocity Array
W(4)	-	Weight Array
X(15)	-	State Vector

Table A2: BASIC (FORTRAN) Variables

A$, B$, C$, D$, LOOP$	-	General Character Strings
ADOT	-	Rate of Change of Angle of Attack <rad/s>
ANGLE	-	Output Angle of the ArcTangent Function
ANS$ (BZ)	-	General Answer <Character String>
AOA	-	Angle of Attack <rad>
AOP	-	Argument of Pericynthion <rad>
APG	-	Apocynthion Radius <ft>
APGH	-	Former (Hold) Apocynthion Value <ft>
AZH	-	Heading Azimuth <rad>
COTG	-	CoTarget (Node Opposite Insertion) Altitude <nm>
DENOM	-	Denominator for the ArcTangent Function <n.d.>
DG	-	Change in Flight Path Angle (Gamma) <rad>
DI$ (--)	-	Input Drive Letter <Character>
DR	-	Change in MECO Time (Runtime) <s>
DT0	-	Initial Step Size <1 second>
DT	-	Step Size <1 second>
DV	-	Performance Delta Velocity <ft/s>
DV2	-	Velocity Change for Insertion/Deorbit <ft/s>
DX$ (--)	-	Output Drive Letter <Character>
ECA	-	Eccentric Anomaly <rad>
ECC	-	Eccentricity <n.d.>
G0	-	Gravity at the Lunar Surface <5.31 ft/s >
GAM0	-	Former MECO Flight Path Angle <rad>

GAMFLAG (IGAMFLAG) - Flight Path Angle Iteration Counter <Integer>

GAMH - Former Pitch-over Angle <rad>

GAMI - Inertial Flight Path Angle <rad>

GAML - Local Flight Path Angle <rad>

GAMP - Pitch-over Angle <rad>

GAMT - Thrust Elevation Angle <rad>

GE - Gravity at the Earth's Surface <32.2 ft/s >

H - Altitude <ft>

HEAD0 - Initial Heading <rad>

HEAD - Actual Heading <rad>

HEADD - Actual Heading <deg>

HEADT - Thrust Heading Angle <rad>

I, J, K, TEMP, TEMP1, TEMP2, TEMP3 - General Variables

IFLAG - Inertial Print Marker <0 - off, 1 - on>

INCL - Inclination <rad>

INCLN - Inclination <rad>

ITER - Iteration Print Counter <Integer>

LAN - Longitude of the Ascending Node <rad>

LATD - Latitude of the Spacecraft <deg>

LONGD - Longitude of the Spacecraft <deg>

(LNS) - Number of Output Data Lines <Integer>

MD1, MDOT - Rate of Fuel Use <slug/s>

MEA - Mean Eccentric Anomaly <rad>

MF - Final Mass after Insertion/before Deorbit <slugs>

MFU - Fuel Mass <slug>

MFUEL - Insertion/Deorbit Fuel Requirement <slugs>

MU - Lunar Gravitational Parameter <1.73x10^{14} ft^3/s^2>

NOF$ (NOFZ) - Name Of File <Character String>

NUMOR - Numerator for the ArcTangent Function

OMEGA - Rotation Rate of the Moon <2.6622E-6 rad/s>

ORFLAG (IORFLAG) - Orbit Calculation Marker <1 - on, 0 - off>

OUTFLAG (IOUTFLAG) - Output Control Marker <Integer>

OUTINT - Time Between Outputs <s>

OUTTIME (OUTTIM) - Time of Output Printing <s>

PDOT - Angular Rate of Latitude <rad/s>

PEG - Pericynthion Radius <ft>

PFLAG (IPFLAG) - Pitch-over Angle <rad>

PHI0 - Angular rate of Latitude of Landing Site <rad/s>

PI - 3.14159 <rad/semicircle>

PPAD - Latitude of Landing Site <rad>

R0 - Radial Distance to Landing Site <ft>

RANGE - Groundtrack Distance to the Landing Site <ft>

RANGE2 - North Range Groundtrack Distance <ft>

RANGE3 - East Range Groundtrack Distance <ft>

RDA - Tangential Speed of the Atmosphere <ft/s>

RDOT - Rate of Radial Distance Change <ft/s>

PRF1, PRF - Normalized Thrust Level <n.d.>

Q1$ - Trajectory Print Marker <Character>

RTFLAG (IRTFLAG) - MECO Time Iteration Counter <Integer>

RTH - Former MECO Time <s>

RUNTIME (RUNTIM) - Simulation Stop Time <s>

SLR - Simi-latus Rectum <ft>

SMJ - Simi-major Axis <ft>

T1, T - Thrust <lbf>

TDOT - Angular Rate of Longitude <rad/s>

TGT - Target (Boost Orbit Apocynthion) Altitude <nm>

THETA - Output Angle for Cosine Function <rad>

THETA0 - Initial Longitude of Landing Site <rad>

TIME, TTEMP (TIM) - Simulation Time <s>

TPAD - Longitude of the Landing Pad <rad>

TTOW - Thrust to Weight <G's>

TYP$ (TYPZ) - Simulation Type ('A' or 'D') <Character String>

V2 - Local Tangential Velocity Component <ft/s>

V - Local Speed of the Spacecraft <ft/s>

WEIGHT - Spacecraft Weight <lb>

WPLD - Payload Weight <lbf>

X - Temporary Variable for ArcCosine Function - <n.d.>

APPENDIX B: Program Listings

BASIC Version

```
100  ' ------------------------------------------------------------------
110  '|                        LANDER MAIN PROGRAM                        |
120  '|-----------------------------------------------------------------|
130  '|***  TITLE       :  Lunar Lander Trajectory Simulation             |
140  '|     NAME        :  LANDER.BAS                                     |
150  '|     AUTHOR      :  Chris Varner                                   |
160  '|     FOR         :  Lunar Base Systems Study (LBSS)                |
170  '|     DATE        :  22 June, 1988                                  |
180  '|                                                                   |
190  '|***  PURPOSE:  The phase of flight between lunar orbit and the sur- |
200  '|               face can not be approximated using ideal free space  |
210  '|               equations.  The lunar lander trajectory simulation   |
220  '|               is used to analyze the flight characteristics and the|
230  '|               control requirements necessary for a descent to the  |
240  '|               lunar surface.                                        |
250  '|                                                                   |
260  '|***  NOTES:  Refer to the LANDER Program Manual for specific infor- |
270  '|             mation on operation of this program.                   |
280  '|                                                                   |
290  '|***  VARIABLES:  DT0      =  Initial Time step (1 second)           |
300  '|                 DX$      =  Output Drive Letter <Character>        |
310  '|                 NOF$     =  Name Of File <Character String>        |
320  '|                 ORFLAG   =  Orbit Calculation Flag ("0"-off: "1"-on)|
330  '|                 OUTTIME  =  Time of next output <s>                |
340  '|                 TIME     =  Time of simulation <s>                 |
350  '|                                                                   |
360  ' ------------------------------------------------------------------
1000 '******************************
1010 '*** User Defined Functions ***
1020 '******************************
1030  PI = 4 * ATN(1)
1040  DEF FNARCCOS (X) = -ATN(X / SQR(-X * X + 1)) + PI / 2
1050 '************************
1060 '*** Dimension Arrays ***
1070 '************************
1080  DIM C(3, 3), K1(15), K2(15), K3(15), K4(15), M(4), PSN(3), RKX(15)
1090  DIM RKDX(15), SDAT(5), TRAJDAT(100, 20), VEL(3), VSP(3), W(4), X(15)
1100 '******************
1110 '*** Data Entry ***
1120 '******************
1130 ' DX$ = "D"
1140  'GOTO 1870
1150  GOSUB 10000                       'Data Entry
1160  ITER = 0
1170  RTFLAG = 1
1180 '***********************************
1190 '*** Begin Burn Time Iteration ***
```

```
1200 '*********************************
1210    GAMFLAG = 1
1220 '  ****************************************
1230 '  *** Begin Flight Path Angle Iteration ***
1240 '  ****************************************
1250 '    ******************************
1260 '    *** Variable Initialization ***
1270 '    ******************************
1280      GOSUB 12000                       'Variable Initialization
1290      NOF$ = DX$ + ":LOUTPUT.PRN"
1300      OPEN "O", #3, NOF$
1310 '    *******************************************
1320 '    *** Start the Iteration/Integration Loop ***
1330 '    *******************************************
1340         TIME = INT(TIME * 100) / 100
1350 '      ****************
1360 '      *** Integrate ***
1370 '      ****************
1380        GOSUB 13000                     'Runge-Kutta 4
1450 '      ****************************************************************
1460 '      *** Continue iteration sequence until the simulation time    ***
1470 '      *** exceeds the desired stop time (RUNTIME).                 ***
1480 '      ****************************************************************
1490      IF TIME < RUNTIME THEN 1340
1500 '    ***********************************
1510 '    *** Determine Orbital Parameters ***
1520 '    ***********************************
1530      GOSUB 15000                       'Orbit
1540 '    ***********************************
1550 '    *** Print the Orbital Parameters ***
1560 '    ***********************************
1570      ORFLAG = 1
1580      GOSUB 14000                       'Output
1590      CLOSE #3
1592 '    ***********************************************
1594 '    *** Iterate the Pitch-over Flight Path Angle ***
1595 '    ***********************************************
1600      IF GAMFLAG > 1 THEN 1640
1610        GAMH = GAMP
1620        GAMP = GAMP + 2 * PI / 180
1630        GOTO 1680
1640 '    ELSE
1650        TEMP = GAMP
1660        DG = (GAMP - GAMH) * (0 - GAMI) / (GAMI - GAM0)
1662        IF DG > 5 THEN DG = 5
1664        IF DG < -5 THEN DG = -5
1667        GAMP = GAMP + DG
```

```
1670         GAMH = TEMP
1680 '     ENDIF
1690       GAM0 = GAMI
1700       GAMFLAG = GAMFLAG + 1
1710       OUTFLAG = 1
1720       GOSUB 14000                          'Output
1730       CLOSE #3
1740     IF ABS(GAMI) > .01 * PI / 180 THEN 1280
1742 '   *****************************
1744 '   *** Iterate the MECO Time ***
1745 '   *****************************
1750     IF RTFLAG > 1 THEN 1790
1760       RTH = RUNTIME
1770       RUNTIME = RUNTIME + 2
1780       GOTO 1830
1790 '   ELSE
1800       TEMP = RUNTIME
1810       DR = (RUNTIME - RTH) * (TGT - APG) / (APG - APGH)
1812       IF DR > 20 THEN DR = 20
1813       IF DR < -20 THEN DR = -20
1815       RUNTIME = RUNTIME + DR
1820       RTH = TEMP
1830 '   ENDIF
1840     APGH = APG
1850     RTFLAG = RTFLAG + 1
1860   IF ABS(RTH - RUNTIME) > 1 AND X(4) > M(1) THEN 1210
1861 '**************************
1862 '*** Print Final Output ***
1863 '**************************
1865   IF X(4) <= M(1) THEN 1985
1870     NOF$ = DX$ + ":LOUTPUT.PRN"
1880     OPEN "I", #3, NOF$
1890     TEMP = 0
1900       TEMP = TEMP + 1
1910       FOR I = 1 TO 18
1920         IF EOF(3) THEN 1940
1930           INPUT #3, TRAJDAT(TEMP, I)
1940 '       ENDIF
1950       NEXT I
1960     IF EOF(3) THEN 1970 ELSE 1900
1970     OUTFLAG = 2
1980     GOSUB 14000                          'Output
1982     GOTO 1987
1985 'ELSE
1986     PRINT "*** Not Enough Propellant ***"
1987 'ENDIF
1990   KEY ON
```

```
2000 END
10000 ' -------------------------------------------------------------
10010 '|                     Data Entry Subroutine                  |
10020 '|------------------------------------------------------------|
10030 '|                                                            |
10040 '|Name   : DE                                                 |
10050 '|Author : Chris Varner                                       |
10060 '|Date   : 30 December, 1986                                  |
10070 '|                                                            |
10080 '|*** Purpose:  This routine is used to enter the data required for|
10090 '|              program operation.                            |
10100 '|                                                            |
10110 ' -------------------------------------------------------------
10120  CLS : KEY OFF
10125  LOCATE 10, 1
10130  INPUT "Drive for Input data files -----------------------  ", DI$
10140  INPUT "Drive for Output data files ----------------------  ", DX$
10150  LOOP$ = "ON"
10160    INPUT "Choose 'F' for File Entry or 'M' for Manual Entry.  ", ANS$
10170    IF ANS$ = "M" OR ANS$ = "m" THEN LOOP$ = "OFF"
10180    IF ANS$ = "F" OR ANS$ = "f" THEN LOOP$ = "OFF"
10190  IF LOOP$ = "ON" THEN 10160
10200  LOOP$ = "ON"
10210    INPUT "Is this to be an Ascent or a Descent simulation?    ", TYP$
10220    IF LEFT$(TYP$, 1) = "A" THEN TYP$ = "A": LOOP$ = "OFF"
10230    IF LEFT$(TYP$, 1) = "a" THEN TYP$ = "A": LOOP$ = "OFF"
10240    IF LEFT$(TYP$, 1) = "D" THEN TYP$ = "D": LOOP$ = "OFF"
10250    IF LEFT$(TYP$, 1) = "d" THEN TYP$ = "D": LOOP$ = "OFF"
10260  IF LOOP$ = "ON" THEN 10210
10270  CLS
10280  PRINT
10290  PRINT
10300  PRINT
10310  PRINT "Lunar Landing Site"
10320  PRINT "Landing Site Latitude    (-90 to +90)    "
10330  PRINT "Landing Site Longitude   (0 to 360)      "
10340  PRINT
10350  PRINT "*****  Vehicle Configuration  *****"
10360  PRINT
10370  PRINT "                      Payload Weight <lb>"
10380  PRINT
10390  PRINT
10400  PRINT
10410  PRINT
10420  PRINT
10430  PRINT "--------------------------------------------------------------"
10440  PRINT
```

```
10450  PRINT
10460  PRINT
10470  PRINT "Inert Weight    <lb>                          |Propellant Weight <lb>"
10480  PRINT "Thrust          <lbf>                         |Specific Impulse  <s>"
10490  PRINT "Hover Time       <s> "
10500  LOCATE 5, 45: INPUT "", PHI0
10510  IF PHI0 < -90 OR PHI0 > 90 THEN 10500
10520  LOCATE 6, 45: INPUT "", THETA0
10530  IF THETA0 < 0 OR THETA0 > 360 THEN 10520
10540  IF ANS$ = "F" OR ANS$ = "f" THEN 10680
10550    LOCATE 20, 30: INPUT "", SDAT(1)
10560    LOCATE 20, 70: INPUT "", SDAT(2)
10570    LOCATE 21, 30: INPUT "", SDAT(3)
10580    LOCATE 21, 70: INPUT "", SDAT(4)
10590    LOCATE 22, 30: INPUT "", SDAT(5)
10600    NOF$ = DX$ + ":LANDER.DAT"
10610    OPEN "O", #1, NOF$
10620    PRINT #1, SDAT(1)
10630    PRINT #1, SDAT(2)
10640    PRINT #1, SDAT(3)
10650    PRINT #1, SDAT(4)
10660    PRINT #1, SDAT(5)
10670    GOTO 10760
10680 'ELSE
10690    NOF$ = DI$ + ":LANDER.DAT"
10700    OPEN "I", #1, NOF$
10710    INPUT #1, SDAT(1)
10720    INPUT #1, SDAT(2)
10730    INPUT #1, SDAT(3)
10740    INPUT #1, SDAT(4)
10750    INPUT #1, SDAT(5)
10760 'ENDIF
10770  CLOSE #1
10780  LOCATE 10, 48: INPUT "", WPLD
10790  CLS
10795  LOCATE 1, 45: PRINT "<s>": LOCATE 1, 1
10800  INPUT "Time to Main Engine Cut-off (MECO) ?  ", ANS$
10810  RUNTIME = VAL(ANS$)
10820    LOCATE 2, 1: PRINT "Holding Orbit (      <nm> X       <nm>)        "
10850    LOCATE 2, 17: INPUT "", TGT
10860    LOCATE 10, 20: PRINT "
10870    LOCATE 11, 20: PRINT "                                                  "
10880    LOCATE 2, 29: INPUT "", COTG
10890    IF TGT > COTG THEN TEMP = TGT: TGT = COTG: COTG = TEMP
10900    IF TGT >= 15 THEN 10930
10910      LOCATE 10, 20
10915      PRINT "*** The Orbit's Mimimum Pericynthion Altitude ***"
```

```
10920      LOCATE 11, 20
10925      PRINT "***            is 15 nautical miles.          ***"
10930 '  ENDIF
10940  IF TGT < 15 THEN 10820
10970  PRINT
10980  PRINT " The spacecraft will perform a vertical rise (Flight Path Angle"
10990  PRINT " {Gamma} = 90 deg.) for the first few seconds of flight.  At"
11000  PRINT " a relative velocity of 30 ft/s a pitch-over maneuver is"
11010  PRINT " executed; and the vehicle will momentarily thrust along a"
11020  PRINT " flight path defined by the user (Good Value = 70)."
11030  PRINT
11040  INPUT "Flight path angle at pitch-over?                          ", GAMP
11050  GAMP = GAMP * PI / 180
11060  PRINT
11070  INPUT "Holding orbit inclination ? (0 to 360)                    ", INCLN
11075  INPUT "Do you wish to see the trajectory of each iteration ", Q1$
11076  IF LEFT$(Q1$, 1) = "y" OR LEFT$(Q1$, 1) = "Y" THEN Q1$ = "Y" ELSE Q1$
"N"
11080  IF INCLN > (180 - ABS(PHI0)) AND INCLN < (180 + ABS(PHI0)) THEN 11070
11090  IF INCLN < ABS(PHI0) OR INCL > (360 - ABS(PHI0)) THEN 11070
11100  INCLN = INCLN * PI / 180
11110  X = COS(INCLN) / COS(PHI0 * PI / 180)
11120  GOSUB 22000                        'Inverse Cosine
11130  AZH = -(THETA - PI / 2)            'ArcSine
11140  IF INCLN <= PI / 2 THEN HEAD0 = AZH
11150  IF (INCLN > PI / 2) AND (INCLN <= PI) THEN HEAD0 = 2 * PI + AZH
11160  IF INCLN > PI THEN HEAD0 = PI + AZH
11162  IF TYP$ = "D" THEN 11163 ELSE 11169
11163    IF HEAD0 < PI THEN 11164 ELSE 11166
11164      HEAD0 = HEAD0 + PI
11165      GOTO 11168
11166 '  ELSE
11167      HEAD0 = HEAD0 - PI
11168 '  ENDIF
11169 'ENDIF
11170  CLS
11180  PRINT "*** Calculating ***"
11190  NOF$ = DI$ + ":LAUNCH.DAT"
11200  OPEN "O", #1, NOF$
11220  PRINT #1, THETA0, PHI0
11230  FOR J = 1 TO 5
11240    PRINT #1, SDAT(J)
11250  NEXT J
11260  PRINT #1, RUNTIME, TGT, COTG
11270  PRINT #1, HEAD0
11280  PRINT #1, WPLD, GAMP
11290  CLOSE #1
```

```
11300 RETURN
12000 ' -----------------------------------------------------------------------
12010 '|                  Variable Initialization Subroutine                  |
12020 ' -----------------------------------------------------------------------
12030  NOF$ = DI$ + ":LAUNCH.DAT"
12040  OPEN "I", #1, NOF$
12060  INPUT #1, THETA0, PHI0
12070  FOR J = 1 TO 5
12080    INPUT #1, SDAT(J)
12090  NEXT J
12100  INPUT #1, TEMP, TGT, COTG
12110  INPUT #1, HEAD0
12120  INPUT #1, WPLD, TEMP
12130  CLOSE #1
12140  IF TYP$ = "D" THEN SDAT(4) = -SDAT(4)
12142  DT0 = 1
12150  DT = DT0
12160  GAMT = PI / 2
12170  G0 = 1.62 * 3.28084                 'Lunar Surface Gravity <ft/s^2>
12180  GE = 9.810001 * 3.28084             'Terrian Surface Gravity <ft/s^2>
12190  HEAD = HEAD0
12200  IFLAG = 0
12210  MFU = 0
12220  OMEGA = 2.26622E-06                 'Rotation rate of the Moon <Rad/s>
12230  ORFLAG = 0
12235  OUTINT = 5
12240  OUTTIME = -.0001
12250  PDOT = 0
12260  IF TYP$ = "D" THEN RDOT = .5 * 3.28084 ELSE RDOT = 0 'Surface Speed
12270  R0 = 1740000! * 3.28084              'Lunar Radius <ft>
12280  TDOT = OMEGA
12290  TIME = 0
12300  MU = G0 * R0 ^ 2
12310  PHI0 = PHI0 * PI / 180
12320  THETA0 = THETA0 * PI / 180
12330  W(1) = SDAT(1) + WPLD               'dry weight
12340  M(1) = W(1) / GE                    'mass
12350  M(2) = SDAT(2) / GE                 'prop mass
12360  X(8) = M(1)
12370  X(1) = R0
12380  X(2) = THETA0
12390  X(3) = PHI0
12400  X(4) = M(1)
12410  IF TYP$ = "A" THEN X(4) = X(4) + M(2)
12420  X(5) = RDOT
12430  X(6) = TDOT
12440  X(7) = PDOT
```

```
12450 RETURN
13000 ' -----------------------------------------------------------------
13010 '|                   Integration Subroutine (Runge-Kutta 4)           |
13020 ' -----------------------------------------------------------------
13030  TTEMP = TIME
13040  FOR I = 1 TO 7
13050    RKX(I) = X(I)
13060  NEXT I
13070  GOSUB 20000
13072 '*************************************************************
13074 '*** If it is time to output data goto the output subroutine ***
13075 '*************************************************************
13076  IF TIME > OUTTIME THEN 13077 ELSE 13078
13077    GOSUB 14000                    'Output
13078 'ENDIF
13080  FOR I = 1 TO 7
13090    K1(I) = RKDX(I) * DT
13100    RKX(I) = X(I) + .5 * K1(I)
13110  NEXT I
13120  TIME = TIME + .5 * DT
13130  GOSUB 20000
13140  FOR I = 1 TO 7
13150    K2(I) = RKDX(I) * DT
13160    RKX(I) = X(I) + .5 * K2(I)
13170  NEXT I
13180  GOSUB 20000
13190  FOR I = 1 TO 7
13200    K3(I) = RKDX(I) * DT
13210    RKX(I) = X(I) + K3(I)
13220  NEXT I
13230  TIME = TTEMP + DT
13240  GOSUB 20000
13250  FOR I = 1 TO 7
13260    K4(I) = DT * RKDX(I)
13270    X(I) = X(I) + (K1(I) + 2 * K2(I) + 2 * K3(I) + K4(I)) / 6
13280  NEXT I
13290  RETURN
14000 ' -------------------------------------------------------------------
14010 '|                          Output Subroutine                          |
14020 ' -------------------------------------------------------------------
14030  IF OUTFLAG <> 2 THEN 14160
14045    IF TYP$ = "A" THEN 14104
14050      J = TEMP
14060      FOR I = 1 TO INT((TEMP - 1) / 2)
14062        FOR K = 2 TO 18
14064          TEMP1 = TRAJDAT(J - I, K)
14065          TRAJDAT(J - I, K) = TRAJDAT(I, K)
```

```
14066          TRAJDAT(I, K) = TEMP1
14068        NEXT K
14069      NEXT I
14075 PRINT
14076 PRINT "  Weight Prior to Deorbit Burn <lb>        :"; TRAJDAT(TEMP, 11)
14077 PRINT
14078 PRINT "  Delta Velocity Required to Deorbit"
14079 PRINT "  to the Initial Descent Orbit  <ft/s>    :"; TRAJDAT(TEMP, 9)
14080 PRINT
14081 PRINT "  Fuel Required for the Deorbit Burn <lbf>:"; TRAJDAT(TEMP, 10)
14082 PRINT
14083 PRINT "  Initial Descent Orbit:": PRINT
14084 PRINT "    Apocynthion                        <nm>    -- "; TRAJDAT(TEMP, 2)
14085 PRINT "    Pericynthion                       <nm>    -- "; TRAJDAT(TEMP, 3)
14086      TRAJDAT(TEMP, 4) = 180 - TRAJDAT(TEMP, 4)
14087 PRINT "    Inclination                        <deg>   -- "; TRAJDAT(TEMP, 4)
14088      IF TRAJDAT(TEMP, 5) < 180 THEN 14089 ELSE 14091
14089        TRAJDAT(TEMP, 5) = TRAJDAT(TEMP, 5) + 180
14090        GOTO 14093
14091 '    ELSE
14092        TRAJAT(TEMP, 5) = TRAJDAT(TEMP, 5) - 180
14093 '    END IF
14094 PRINT "    Longitude of the Ascending Node <deg>   -- "; TRAJDAT(TEMP, 5)
14095      IF TRAJDAT(TEMP, 6) < 180 THEN 14096 ELSE 14098
14096        TRAJDAT(TEMP, 6) = 180 - TRAJDAT(TEMP, 6)
14097        GOTO 14100
14098 '    ELSE
14099        TRAJDAT(TEMP, 6) = 540 - TRAJDAT(TEMP, 6)
14100 '    END IF
14101 PRINT "    Argument of Pericynthion           <deg>   -- "; TRAJDAT(TEMP, 6)
14102 PRINT "    Eccentricity                       <n.d.> -- "; TRAJDAT(TEMP, 7)
14103 PRINT
14104 '  ENDIF
14105 PRINT "Time  Altitude  Range  Velocity  Gamma  Heading  Thrust  Weight"
14106 PRINT "<s>      <ft>    <nm>    <ft/s>   <deg>   <deg>   <lbf>    <lb>"
14107    FOR I = 1 TO TEMP - 1
14108      TIME$ = "00:00:00":
14109      WHILE TIME$ < "00:00:01": WEND
14118 PRINT USING "####  #######"; TRAJDAT(I, 1); TRAJDAT(I, 2);
14119 PRINT USING "#######   ######"; TRAJDAT(I, 3); TRAJDAT(I, 6);
14120 PRINT USING "######.##  ###.##"; TRAJDAT(I, 8); TRAJDAT(I, 18);
14121 PRINT USING "######### #######"; TRAJDAT(I, 11); TRAJDAT(I, 12)
14122    NEXT I
14131 PRINT "Ideal Performance Delta Velocity is "; TRAJDAT(TEMP, 8); " <ft/s
14132    IF TYP$ = "D" THEN 14153
14133 PRINT
14134 PRINT "  Orbit Attained:"
```

```
14135 PRINT "    Apocynthion                       <nm>    -- "; TRAJDAT(TEMP, 2)
14136 PRINT "    Pericynthion                      <nm>    -- "; TRAJDAT(TEMP, 3)
14137 PRINT "    Inclination                       <deg>   -- "; TRAJDAT(TEMP, 4)
14138 PRINT "    Longitude of the Ascending Node <deg>   -- "; TRAJDAT(TEMP, 5)
14139 PRINT "    Argument of Pericynthion          <deg>   -- "; TRAJDAT(TEMP, 6)
14140 PRINT "    Eccentricity                      <n.d.> -- "; TRAJDAT(TEMP, 7)
14141 PRINT
14142 PRINT "  Velocity Required at Apocynthion to"
14143 PRINT "  Achieve the Holding Orbit              <ft/s>:"; TRAJDAT(TEMP, 10)
14144 PRINT
14149 PRINT "  Fuel Required for the Apocynthion Burn<lb>:"; -TRAJDAT(TEMP, 11
14150 PRINT
14151 PRINT "  Weight After Apocynthion Burn          <lb>: "; TRAJDAT(TEMP, 12
14152 PRINT "  Weight of the Payload Placed in Orbit <lb>: "; TRAJDAT(TEMP, 9)
14153 '   ENDIF
14154     PRINT
14155     PRINT "********* Simulation Complete ***********"
14156     OUTFLAG = 0
14157     GOTO 14760
14160 'ELSE
14170     IF OUTFLAG = 0 THEN 14250
14180       ITER = ITER + 1
14190       PRINT "Iteration # "; ITER;
14200       PRINT "       Apocynthion = "; APG;
14210       PRINT " <nm>   Pericynthion = "; PEG;
14220       PRINT " <nm>"
14230       OUTFLAG = 0
14240       GOTO 14750
14250 '   ELSE
14260       IF ORFLAG = 1 THEN 14270 ELSE 14550
14270         PRINT #3,
14280         PRINT #3, APG
14290         PRINT #3, PEG
14300         PRINT #3, INCL * 180 / PI
14310         PRINT #3, LAN
14320         PRINT #3, AOP
14330         PRINT #3, ECC
14340         IF TYP$ = "A" THEN 14370
14350           DV = -SDAT(4) * GE * LOG(X(4) / M(1))
14360           GOTO 14390
14370 '       ELSE
14380           DV = SDAT(4) * GE * LOG((M(1) + M(2)) / X(4))
14390 '       ENDIF
14400         PRINT #3, DV
14410 '        PRINT
14420 TEMP = 2 / (R0 + APG * 6076.1) - 2 / (2 * R0 + (APG + COTG) * 6076.1)
14430         DV2 = SQR(MU * TEMP)
```

```
14432 TEMP = 2 / (R0 + APG * 6076.1) - 2 / (2 * R0 + (APG + PEG) * 6076.1)
14434        DV2 = DV2 - SQR(MU * TEMP)
14440        MF = X(4) * EXP(-DV2 / SDAT(4) / GE)
14450        MFUEL = MF - X(4)
14460        IF TYP$ = "D" THEN 14490
14470          WPLD = MF * GE - SDAT(1)
14480          PRINT #3, WPLD
14490 '      ENDIF
14500        PRINT #3, DV2
14510        PRINT #3, MFUEL * GE
14520        PRINT #3, MF * GE
14530        ORFLAG = 0
14540        GOTO 14740
14550 '    ELSE
14560        TPAD = THETA0 + OMEGA * TIME
14570        PPAD = PHI0
14590        RANGE2 = 940 * (X(2) - TPAD)     ' in nautical miles
14600        RANGE3 = 940 * (X(3) - PPAD)       ' in nautical miles
14610        X = COS(X(2) - TPAD) * COS(X(3) - PPAD)
14620        GOSUB 22000                        'Arccosine
14630        RANGE = THETA * 940
14640        LONGD = (X(2) - OMEGA * TIME) * 180 / PI: LATD = X(3) * 180 / PI
14641        IF Q1$ = "N" THEN 14644
14642          U$ = "####.## ######### ###### ###.## ######## ########"
14643          PRINT USING U$; TIME, H, V, GAML * 180 / PI, T, WEIGHT
14644 '      ENDIF
14645        IF TYP$ = "D" THEN 14646 ELSE 14652
14646          IF HEADD < 180 THEN 14647 ELSE 14649
14647            HEADD = HEADD + 180
14648            GOTO 14651
14649 '        ELSE
14650            HEADD = HEADD - 180
14651 '        ENDIF
14652 '      ENDIF
14653 PRINT #3, TIME, H, RANGE, LONGD;
14660 PRINT #3, USING " ####.## ###### ######"; LATD, V, VSP(0);
14670 PRINT #3, USING " ###.## ###.## ###.##"; GAML * 180 / PI, GAMI * 180
PI, AOA;
14680 PRINT #3, USING " ########  ########"; T, WEIGHT;
14690 PRINT #3, USING " ###.##  ######  ######"; ADOT, RANGE, RANGE2;
14700 PRINT #3, USING " ###### ##.## ###.##"; RANGE3, TTOW, HEADD
14710          OUTTIME = OUTTIME + OUTINT
14720          OUTTIME = INT(OUTTIME)
14730        IF OUTTIME < TIME THEN 14710
14740 '    ENDIF
14750 '  ENDIF
14760 'ENDIF
```

```
14770 RETURN
15000 ' --------------------------------------------------------------------
15010 '|                    Orbital Parameters Subroutine                    |
15020 ' --------------------------------------------------------------------
15030  C(1, 1) = COS(X(3)) * COS(X(2)): C(1, 2) = COS(X(3)) * SIN(X(2))
15040  C(1, 3) = SIN(X(3))
15050  C(2, 1) = -SIN(X(2)): C(2, 2) = COS(X(2)): C(2, 3) = 0
15060  C(3, 1) = -SIN(X(3)) * COS(X(2)): C(3, 2) = -SIN(X(3)) * SIN(X(2))
15070  C(3, 3) = COS(X(3))
15080  FOR I = 1 TO 3
15090    TEMP = 0
15100    FOR J = 1 TO 3
15110      TEMP = TEMP + VSP(J) * C(J, I)
15120    NEXT J
15130    VEL(I) = TEMP
15140  NEXT I
15150  PSN(1) = X(1) * COS(X(3)) * COS(X(2))
15160  PSN(2) = X(1) * COS(X(3)) * SIN(X(2))
15170  PSN(3) = X(1) * SIN(X(3))
15180  PSN(0) = SQR(PSN(1) ^ 2 + PSN(2) ^ 2 + PSN(3) ^ 2)
15190  VEL(0) = SQR(VEL(1) ^ 2 + VEL(2) ^ 2 + VEL(3) ^ 2)
15200  SMJ = 1 / (2 / PSN(0) - VEL(0) ^ 2 / MU)
15210  I = PSN(1) * VEL(1) + PSN(2) * VEL(2) + PSN(3) * VEL(3)
15220  J = 1 - PSN(0) / SMJ
15230  K = I / SQR(MU * SMJ)
15240  ECC = SQR(J ^ 2 + K ^ 2)
15250  NUMOR = K: DENOM = J: GOSUB 21000     'ArcTan360
15260  ECA = ANGLE
15270  MEA = ECA - ECC * SIN(ECA)
15280  K = MU * J / PSN(0)
15290  C(1, 1) = (K * PSN(1) - I * VEL(1)) / MU / ECC
15300  C(1, 2) = (K * PSN(2) - I * VEL(2)) / MU / ECC
15310  C(1, 3) = (K * PSN(3) - I * VEL(3)) / MU / ECC
15320  SLR = SMJ * (1 - ECC ^ 2)
15330  J = PSN(0) - SLR
15340  K = I / PSN(0)
15350  C(2, 1) = (K * PSN(1) - J * VEL(1)) / ECC / SQR(MU * SLR)
15360  C(2, 2) = (K * PSN(2) - J * VEL(2)) / ECC / SQR(MU * SLR)
15370  C(2, 3) = (K * PSN(3) - J * VEL(3)) / ECC / SQR(MU * SLR)
15380  C(3, 1) = C(1, 2) * C(2, 3) - C(1, 3) * C(2, 2)
15390  C(3, 2) = C(1, 3) * C(2, 1) - C(1, 1) * C(2, 3)
15400  C(3, 3) = C(1, 1) * C(2, 2) - C(1, 2) * C(2, 1)
15410  X = C(3, 3)
15420  GOSUB 22000                          'ARCCOSIGN
15430  INCL = THETA
15440  NUMOR = C(3, 1): DENOM = -C(3, 2): GOSUB 21000  'ArcTan360
15450  LAN = ANGLE
```

```
15460  LAN = LAN * 180 / 3.141592654#
15470  NUMOR = C(1, 3): DENOM = C(2, 3): GOSUB 21000  'ArcTan360
15480  AOP = ANGLE
15490  AOP = AOP * 180 / 3.141592654#
15500  APG = (SMJ * (1 + ECC) - R0) / 6078
15510  PEG = (SMJ * (1 - ECC) - R0) / 6078
15520 RETURN
20000 ' ----------------------------------------------------------------------
20010 '|                         Equations of Motion                          |
20020 ' ----------------------------------------------------------------------
20030 '********************************
20040 '*** Preliminary Calculations ***
20050 '********************************
20060  DT = DT0
20080  RKX(8) = RKX(4)
20090  VSP(1) = RKX(5)
20100  VSP(2) = RKX(1) * RKX(6) * COS(RKX(3))
20110  VSP(3) = RKX(1) * RKX(7)
20120  VSP(0) = SQR(VSP(1) ^ 2 + VSP(2) ^ 2 + VSP(3) ^ 2)
20130  RDA = R0 * OMEGA * COS(PHI0)
20140  V2 = VSP(2) - RDA
20150  V = SQR(VSP(1) ^ 2 + V2 ^ 2 + VSP(3) ^ 2)
20160  H = RKX(1) - R0
20170  GAMI = ATN(VSP(1) / SQR(VSP(2) ^ 2 + VSP(3) ^ 2))
20180  IF V2 = 0 AND VSP(3) = 0 THEN 20190 ELSE 20220
20190    GAML = 90 * PI / 180
20200    GOTO 20230
20210 'ELSE
20220    GAML = ATN(VSP(1) / SQR(V2 ^ 2 + VSP(3) ^ 2))
20230 'ENDIF
20240  NUMOR = RKX(7)
20250  DENOM = RKX(6) - OMEGA
20260  GOSUB 21000                       'ArcTan 360
20270  IF ANGLE <= PI / 2 THEN HEAD = PI / 2 - ANGLE
20280  IF ANGLE > PI / 2 THEN HEAD = 5 * PI / 2 - ANGLE
20290  GOSUB 24000                     'Control
20300  GOSUB 23000                     'PROFILE
20310  T1 = SDAT(3) * PRF
20320  MD1 = -T1 / GE / SDAT(4)
20330  T = T1
20340  HEADD = HEAD * 180 / PI
20350  WEIGHT = RKX(4) * GE
20360  TTOW = T / WEIGHT
20370  MDOT = MD1
20380  FOR I = 1 TO 7
20390    RKX(I + 8) = RKX(I)
20400  NEXT I
```

```
20410 '*****************************************************
20420 '*** Equations of Motion for Spherical Coordinates ***
20430 '*****************************************************
20440 RKDX(1) = RKX(5)
20450 RKDX(2) = RKX(6)
20460 RKDX(3) = RKX(7)
20470 RKDX(4) = MDOT
20480 TEMP1 = 0: TEMP2 = 0: TEMP3 = 0
20490 TEMP1 = T * SIN(GAMT)
20500 TEMP1 = TEMP1 / RKX(4)
20510 TEMP2 = (T * COS(GAMT) * SIN(HEADT)) / (RKX(4) * RKX(1) * COS(RKX(3)))
20520 TEMP3 = (T * COS(GAMT) * COS(HEADT)) / (RKX(4) * RKX(1))
20530 RKDX(5) = RKX(1) * RKX(6) ^ 2 * COS(RKX(3)) ^ 2 + RKX(1) * RKX(7) ^ 2
20540 RKDX(5) = RKDX(5) - MU / RKX(1) ^ 2 + TEMP1
20550 RKDX(6) = 2 * (-RKX(5) * RKX(6) / RKX(1) + RKX(6) * RKX(7) * TAN(RKX(3))
20560 RKDX(6) = RKDX(6) + TEMP2
20570 RKDX(7) = -2 * RKX(5) * RKX(7) / RKX(1)
20580 RKDX(7) = RKDX(7) - RKX(6) ^ 2 * SIN(RKX(3)) * COS(RKX(3)) + TEMP3
20590 RETURN
21000 ' ----------------------------------------------------------------
21010 '|                    ArcTan360 Function                          |
21020 ' ----------------------------------------------------------------
21030 '
21040  IF NUMOR > 0 AND DENOM = 0 THEN 21050 ELSE 21070
21050    ANGLE = 3.141592654# / 2
21060    GOTO 21240
21070 'ELSE
21080    IF DENOM = 0 THEN 21090 ELSE 21110
21090      ANGLE = -3.141592654# / 2
21100      GOTO 21230
21110 '  ELSE
21120      IF NUMOR >= 0 AND DENOM > 0 THEN 21130 ELSE 21150
21130        ANGLE = ATN(NUMOR / DENOM)
21140        GOTO 21220
21150 '    ELSE
21160        IF NUMOR < 0 AND DENOM > 0 THEN 21170 ELSE 21190
21170          ANGLE = 2 * 3.141592654# + ATN(NUMOR / DENOM)
21180          GOTO 21210
21190 '      ELSE
21200          ANGLE = 3.141592654# + ATN(NUMOR / DENOM)
21210 '      ENDIF
21220 '    ENDIF
21230 '  ENDIF
21240 'ENDIF
21250  RETURN
22000 ' -----------------------------------------------------------------
22010 '|                    ArcCosine Function                           |
```

```
22020 ' ----------------------------------------------------------------------
22030 '|***   TITLE  :  Calculation of ArcCosine
22040 '|      NAME   :  ARCCOS
22050 '|      AUTHOR:   Chris Varner
22060 '|      FOR    :  Personal Library
22070 '|      DATE   :  22 August, 1986
22080 '|
22090 '|**   PURPOSE:   Outputs the ArcCosine of X as THETA.
22100 '|
22110 '|**   NOTES   : Define the function:
22120 '|               FNARCCOS(X) = -ATN(X/SQR(-X*X + 1)) + 3.141592654/2
22130 '|               at the beginning of the main program.
22140 '|
22150 '|** VARIABLES:   THETA    - The ArcCosine of X <rad>
22160 '|                X        - The adjacent/hypotenuse <n.d.>
22170 '|
22180 '|** RESERVED VARIABLES:     PI
22190 '|
22200 '|
22210 ' ----------------------------------------------------------------------
22220 '*** Define PI
22230 PI = 3.141592654#
22240 '*** Test for singularities in the derived function.
22250 IF X > 1 THEN X = 1
22260 IF X < -1 THEN X = -1
22270 IF X = 1 THEN 22280 ELSE 22300
22280   THETA = 0
22290   GOTO 22350
22300 IF X = -1 THEN 22310 ELSE 22340
22310   THETA = PI
22320   GOTO 22350
22330 '*** If there are no singularities, calculate arccosine of X.
22340 THETA = FNARCCOS(X)
22350 'END ARCCOSINE
22360 RETURN
23000 ' ---------------------------------------------------------------------
23010 '|                    Thrust Profile Subroutine                         |
23020 ' ---------------------------------------------------------------------
23030  IF TYP$ = "A" THEN 23170
23040    IF TIME <= SDAT(5) THEN 23120
23050      IF TIME > SDAT(5) + 35 THEN 23080
23060        PRF = PRF1 + (1 - PRF1) / 35 * (TIME - SDAT(5))
23070        GOTO 23100
23080 '    ELSE
23090        PRF = 1
23100 '    ENDIF
23110      GOTO 23150
```

```
23120 '  ELSE
23130      PRF = RKX(4) * G0 / SDAT(3)
23140      PRF1 = PRF
23150 '  ENDIF
23160    GOTO 23200
23170 'ELSE
23180    PRF = 1
23190    PRF1 = 1
23200 'ENDIF
23210  RETURN
24000 ' ------------------------------------------------------------------
24010 '|                Control Procedures Subroutine                      |
24020 '|------------------------------------------------------------------|
24030 '|*** TITLE       :  Control Procedures                              |
24040 '|    NAME        :  LCONTROL                                        |
24050 '|    AUTHOR      :  Chris Varner                                    |
24060 '|    FOR         :  LAUNCH PROGRAM                                  |
24070 '|    DATE        :  15 June, 1987                                   |
24080 '|                                                                   |
24090 '|*** PURPOSE:  Provides control and guidance for Eagle's Ascent     |
24100 '|              Program.  The method of control is that of a zero    |
24110 '|              angle of attack trajectory turn. (Gravity Turn).     |
24120 '|                                                                   |
24130 ' ------------------------------------------------------------------
24140    IF V < 30 THEN 24150 ELSE 24210
24150      GAMT = PI / 2
24160 '    GAML = PI / 2
24170      HEADT = HEAD0
24180 '    HEAD = HEAD0
24190      PFLAG = 0
24200      GOTO 24470
24210 '  ELSE
24220      IF PFLAG < 20 THEN 24230 ELSE 24350
24230        IF PFLAG > 10 THEN 24300
24240          GAMT = GAMP - (90 * PI / 180 - GAMP) / 10 * PFLAG
24250 '        GAML = GAMT
24260 '        HEAD = HEAD0
24270          HEADT = HEAD0
24280          PFLAG = PFLAG + 1
24290          GOTO 24330
24300 '      ELSE
24310          GAMT = GAMP + (GAML - GAMP) / 10 * (PFLAG - 10)
24320          PFLAG = PFLAG + 1
24330 '      ENDIF
24340        GOTO 24460
24350 '    ELSE
24360          GAMT = GAML
```

```
24370            IF GAMT < 0 THEN GAMT = 0
24380            IF GAML > 80 * PI / 180 THEN 24390 ELSE 24430
24390              HEADT = HEAD0
24400              HEAD = HEAD0
24410              GOTO 24440
24420 '          ELSE
24430              HEADT = HEAD0
24440 '          ENDIF
24450 '        ENDIF
24460 '      ENDIF
24470 '    ENDIF
24480 'ENDIF
24490 RETURN
```

FORTRAN Version

```
C     ---------------------------------------------------------------
C    |                       LANDER MAIN PROGRAM                      |
C    |---------------------------------------------------------------|
C    |***  TITLE        :  Lunar Lander Trajectory Simulation         |
C    |     NAME         :  LANDER.BAS                                 |
C    |     AUTHOR       :  Chris Varner                               |
C    |     TRANSLATOR   :  Mike D'Onofrio                             |
C    |     FOR          :  Lunar Base Systems Study (LBSS)            |
C    |     DATE         :  15 August 1988                             |
C    |                                                                |
C    |***  PURPOSE:  The phase of flight between lunar orbit and      |
C    |               surface cannot be approximated using ideal      |
C    |               free space equations.  The lunar lander         |
C    |               trajectory simulation is used to analyze the|
C    |               flight characteristics and the control          |
C    |               requirements necessary for a descent to the     |
C    |               lunar surface.                                  |
C    |                                                                |
C    |***  NOTES: Refer to the LANDER Program Manual for specific|
C    |              information on operation of this program.       |
C    |                                                                |
C     ----------------------------------------------------------------
C     *** Declare Variables **
      IMPLICIT REAL *16 (A-Z)
      INTEGER IGAMFLAG, I, IFLAG, ITER, J, K, LNS, IORFLAG, IOUTFLAG
      INTEGER IPFLAG, IRTFLAG
      CHARACTER LOOPZ*3, NOFZ*72, BZ*1, TYPZ*1
C     *** Dimension Arrays ***
      DIMENSION C(3, 3), D6(5), K1(15), K2(15), K3(15), K4(15)
      DIMENSION PSN(3), RKX(15), RKDX(15)
      DIMENSION VEL(4), W(4)
      COMMON/TOTA/AOA, ADOT, AOP, APG, COTG, DT, ECC, FTPNM, G0
      COMMON/TOTB/GAMI, GAML, GAMT, GE, H, HEAD0, HEAD, HEADD
      COMMON/TOTC/IFLAG, AINCL, AINCLN, ITER, ALAN, LNS, M(4), MU
      COMMON/TOTD/OMEGA, IORFLAG, IOUTFLAG, PEG, PHI0, PI, BZ, R0
      COMMON/TOTE/RUNTIM, SDAT(5), T, TGT, THETA0, TIM
      COMMON/TOTF/TRAJDAT (100, 20), TTOW, TYPZ, V, VSP(4), WEIGHT
      COMMON/TOTG/WPLD, X(15)
      A = 1.
      PI = 4.* QATAN(A)
      CALL DE (GAMP, RUNTIM)
      ITER = 0
      IRTFLAG = 1
      X(4) = M(1) + 1
      GAMP = GAMP * PI / 180.
C     *** Begin Burn TIM  Iteration ***
      DO 1890 WHILE (QABS(RTH - RUNTIM ).GT.1. .AND. X(4).GT.M(1))
```

```
      IGAMFLAG = 1
      GAMI = .2
C     *** Begin Flight Path Angle Iteration ***
      DO 1880 WHILE (QABS(GAMI) .GT. 0.1 * PI / 180.)
C       *** Variable Initialization ***
        CALL INITIALIZE (GAMP, X, OUTTIM, OUTINT)
        LNS = 0
        NOFZ = 'LOUTPUT.DAT'
        OPEN (UNIT=10, FILE=NOFZ, STATUS='OLD', ERR=1335,
     +DISPOSE='DELETE')
 1335   CONTINUE
        CLOSE (UNIT=10)
        OPEN (UNIT=10, STATUS='NEW', FILE=NOFZ)
C       *** Start the Iteration/Integration Loop ***
        DO 1340 WHILE (TIM .LT. RUNTIM)
          TIM = QFLOAT ( INT( TIM * 100. )) / 100.
C         *** Integrate ***
          CALL RK4 (DT, GAMP, LNS, OUTTIM, OUTINT, TIM, X)
C         *******************************************************
C         *** Continue iteration sequence until the simulation **
C         *** time exceeds the desired stop time (RUNTIM).     **
C         *******************************************************
 1340   CONTINUE
C       *** Determine Orbital Parameters ***
        CALL ORBIT (VSP, X, AOP, APG, ECC, AINCL, ALAN, PEG, SMJ)
C       *** Print the Orbital Parameters ***
        IORFLAG = 1
        CALL OUTPUT (GAMP,LNS,IORFLAG,IOUTFLAG,OUTTIM,OUTINT)
C       *** Print Final Output Sequence ***
        IOUTFLAG = 1
        CALL OUTPUT (GAMP,LNS,IORFLAG,IOUTFLAG,OUTTIM,OUTINT)
        CLOSE (UNIT=10)
C       *********************************
C       *** Modify the Pitch-over Angle ***
C       ***  Newton-Raphson Iteration   ***
C       *********************************
        IF ( IGAMFLAG .GT. 1 ) THEN
          TEMP = GAMP
          DG = (GAMP - GAMH ) * ( -GAMI ) / ( GAMI - GAM0 )
          IF ( DG .GT. 5.0 ) DG = 5.0
          IF ( DG .LT. -5.0 ) DG = -5.0
          GAMP = GAMP + DG
          GAMH = TEMP
        ELSE
          GAMH = GAMP
          GAMP = GAMP + 2.0 * PI / 180.
        ENDIF
```

```
          GAM0 = GAMI
          IGAMFLAG = IGAMFLAG + 1
 1880   CONTINUE
C       ********************************
C       ***   Modify the MECO Time   ***
C       *** Newton-Raphson Iteration ***
C       ********************************
        IF (IRTFLAG .GT. 1) THEN
          TEMP = RUNTIM
          DR = (RUNTIM - RTH) * (TGT - APG) / (APG - APGH)
          IF (DR .GT. 50.) DR = 50.
          IF (DR .LT. -50.) DR = -50.
          RUNTIM = RUNTIM + DR
          RTH = TEMP
        ELSE
          RTH = RUNTIM
          RUNTIM  = RUNTIM  + 2.
        ENDIF
        APGH = APG
        IRTFLAG = IRTFLAG + 1
 1890 CONTINUE
      IF (X(4) .LE. M(1)) THEN
        PRINT *, '*** Not Enough Propellant ***'
      ELSE
        NOFZ = 'LOUTPUT.DAT'
        OPEN (UNIT=10, STATUS='OLD', FILE=NOFZ)
        DO I=1, LNS
            READ (10, 1925), (TRAJDAT(I, J), J=1, 17)
 1925 FORMAT (1X, F5.0, F8.0, F7.0, 2F7.2, 2F7.0, 3F6.2, 2F9.0,
     +F6.2, 2F7.0, F5.2, F6.2)
        END DO
        READ (10, 1930), (TRAJDAT(LNS+1, J), J=1, 6)
 1930   FORMAT (1X, F7.1, F7.1, F8.2, F8.2, F8.2, F7.4)
        READ (10, 1932), TRAJDAT(LNS+1, 7)
 1932   FORMAT (1X, F6.1)
        IF (TYPZ .EQ. 'A') THEN
          READ (10, 1938), TRAJDAT (LNS+1, 8)
 1938     FORMAT (1X, F6.0)
          READ (10, 1940), (TRAJDAT(LNS+1, J), J=9, 11)
 1940     FORMAT (1X, F8.1, F8.2, F9.0)
        ELSE
          READ (10, 1945), (TRAJDAT(LNS+1, J), J=8, 10)
 1945     FORMAT (1X, F8.1, F8.2, F9.0)
        ENDIF
        CLOSE (UNIT=10)
        NOFZ = 'LRUN.DAT'
        OPEN (UNIT=10, STATUS='NEW', FILE=NOFZ)
```

```
         IOUTFLAG = 2
         CALL OUTPUT (GAMP, LNS, IORFLAG, IOUTFLAG, OUTTIM, OUTINT)
         CLOSE (UNIT=10)
         OPEN (UNIT=10, STATUS='OLD', FILE='LAUNCH.DAT',
      +DISPOSE='DELETE')
         CLOSE (UNIT=10)
       ENDIF
       STOP
       END
C      ----------------------------------------------------------------
C      |                     Data Entry Subroutine                    |
C      |--------------------------------------------------------------|
C      |                                                              |
C      |Name    : DE                                                  |
C      |Athor   : Chris Varner                                        |
C      |Date    : 3 August, 1986                                      |
C      |                                                              |
C      |*** Purpose:  This routine is used to enter the data          |
C      |              required for program operation.                 |
C      |                                                              |
C       ---------------------------------------------------------------
       SUBROUTINE DE (GAMP, RUNTIM)
C
C      *** Declare Variables **
       IMPLICIT REAL *16 (A-Z)
       INTEGER I, J, K
       CHARACTER LOOPZ*3, NOFZ*72, BZ*1, TYPZ*1
C      *** Dimension Arrays ***
       DIMENSION SDAT(5)
       LOOPZ = 'ON'
       DO 2010 WHILE (LOOPZ .EQ. 'ON')
         WRITE (5, 1999)
         READ (6, 2000), TYPZ
 1999 FORMAT (' Is this to be an Ascent or a Descent Simulation ?')
 2000 FORMAT (A1)
         IF (TYPZ .EQ. 'A') LOOPZ = 'OFF'
         IF (TYPZ .EQ. 'a') THEN
           TYPZ = 'A'
           LOOPZ = 'OFF'
         ENDIF
         IF (TYPZ .EQ. 'D') LOOPZ = 'OFF'
         IF (TYPZ .EQ. 'd') THEN
           TYPZ = 'D'
           LOOPZ = 'OFF'
         ENDIF
 2010 CONTINUE
       WRITE (5, 2020)
```

```
2020   FORMAT (' Lunar Landing Site Latitude  (-90 to 90) ')
     PHI0 = 100.
     DO 2035 WHILE (PHI0 .LT. -90. .OR. PHI0 .GT. 90.)
       READ (6, *), PHI0
2035 CONTINUE
     WRITE (5, 2038)
2038    FORMAT (' Landing Site Longitude   (0 to 360) ')
     THETA0 = 400.
     DO 2045 WHILE (THETA0 .LT. 0. .OR. THETA0 .GT. 360.)
       READ (6, *), THETA0
2045 CONTINUE
     WRITE (5, 2050)
2050   FORMAT (/////////////////////////' *****  Vehicle',
    +' Configuration *****'//' -------------------------',
    +'-----------------------'/)
     WRITE (5, 2059)
     READ (6, *), SDAT(1)
2059   FORMAT (' Inert Weight      <lb>')
     WRITE (5, 2064)
     READ (6, *), SDAT(2)
2064   FORMAT (' Propellant Weight <lb>')
     WRITE (5, 2069)
     READ (6, *), SDAT(3)
2069   FORMAT (' Thrust           <lbf>')
     WRITE (5, 2074)
     READ (6, *), SDAT(4)
2074   FORMAT (' Specific Impulse   <s>')
     WRITE (5, 2079)
     READ (6, *), SDAT(5)
2079   FORMAT (' Hover Time         <s>')
     WRITE (5, 2084)
     READ (6, *), WPLD
2084   FORMAT (' Payload Weight    <lb>')
     WRITE (5, 2089)
     READ (6, *), RUNTIM
2089   FORMAT (/////////////////////////
    +' Time to Main Engine Cut-off (MECO) ? <s> ')
     TGT = 0.
     DO 2120 WHILE (TGT .LT. 15.)
       WRITE (5, 2099)
       READ (6, *), TGT
2099     FORMAT (' Holding Orbit Pericynthion <nm>')
       WRITE (5, 2104)
       READ (6, *), COTG
2104     FORMAT (' Holding Orbit Apocynthion  <nm>')
       IF (TGT .GT. COTG) THEN
         TEMP = TGT
```

```
        TGT = COTG
        COTG = TEMP
      ENDIF
      IF (TGT .LT. 15.) THEN
        WRITE (5, 2110)
2110   FORMAT (/'         *** The Orbit''s Minimum Altitude ***'/
    +          '          ***      is 15 nautical miles.    ***'/)
      ENDIF
2120 CONTINUE
     WRITE (5, 2130)
2130 FORMAT (/' The spacecraft will perform a vertical rise',
    +' (Flight Path Angle'/' {Gamma} = 90 deg.) for the first',
    +' few seconds of flight.  At a')
     WRITE (5, 2140)
2140 FORMAT ('  relative velocity of 30 ft/s a pitch-over',
    +' maneuver is executed;'/' and the vehicle will',
    +' momentarily thrust along a flight path')
     WRITE (5, 2144)
2144 FORMAT (' defined by the user (Good Value = 70).'/)
     WRITE (5, 2149)
     READ (6, *), GAMP
2149 FORMAT ('       Flight path angle at pitch-over?    ')
     AINCLN = 361
     DO 2170 WHILE (AINCLN .LT. QABS(PHI0) .OR. AINCLN .GT. 360.
    +     - QABS(PHI0) .OR. (AINCLN .GT. (180. - QABS(PHI0)) .AND.
    +                   AINCLN .LT. (180. + QABS(PHI0))))
       IF (TYPZ .EQ. 'D') THEN
         WRITE (5, 2150)
       ELSE
         WRITE (5, 2159)
       ENDIF
       READ (6, *), AINCLN
2150 FORMAT (/' Holding orbit inclination ? (0 to 360)       ')
2159 FORMAT (/' Desired orbit inclination ? (0 to 360)       ')
2170 CONTINUE
     WRITE (5, 2179)
     READ (6, 2184), BZ
2179 FORMAT (' Do you wish to see the trajectory of each',
    +' iteration?')
2184 FORMAT (A1)
     PRINT *,'         *** Calculating ***'
     NOFZ = 'LAUNCH.DAT'
     OPEN (UNIT=10, STATUS='NEW', FILE=NOFZ)
     WRITE (10, 2190), AINCLN, THETA0, PHI0, TGT, COTG, WPLD,
    +(SDAT(J), J=1, 5), BZ, TYPZ
2190   FORMAT (1X, F6.2, F7.2, F6.2, 2F5.0, F8.2, 3F9.2, F7.2,
    +F5.1,' ', A1,' ', A1)
```

```
      CLOSE (UNIT=10)
      RETURN
      END
C      -----------------------------------------------------------
C     |               Variable Initialization Subroutine          |
C     |-----------------------------------------------------------|
C     |                                                           |
C     |Name    : INITIALIZE                                       |
C     |Athor   : Chris Varner                                     |
C     |                                                           |
C     |*** Purpose:  This routine is used to set the variables to |
C     |              to their initial values prior to entering    |
C     |              the integration loop.                        |
C     |                                                           |
C      -----------------------------------------------------------
      SUBROUTINE INITIALIZE (GAMP, X, OUTTIM, OUTINT)
C     *** Declare Variables **
      IMPLICIT REAL *16 (A-Z)
      INTEGER IFLAG, ITER, LNS, IORFLAG, IOUTFLAG
      CHARACTER NOFZ*72, BZ*1, TYPZ*1
C     *** Dimension Arrays ***
      DIMENSION W(4), X(15)
      COMMON/TOTA/AOA, ADOT, AOP, APG, COTG, DT, ECC, FTPNM, G0
      COMMON/TOTB/GAMI, GAML, GAMT, GE, H, HEAD0, HEAD, HEADD
      COMMON/TOTC/IFLAG, AINCL, AINCLN, ITER, ALAN, LNS, M(4), MU
      COMMON/TOTD/OMEGA, IORFLAG, IOUTFLAG, PEG, PHI0, PI, BZ, R0
      COMMON/TOTE/RUNTIM, SDAT(5), T, TGT, THETA0, TIM
      COMMON/TOTF/TRAJDAT(100, 20), TTOW, TYPZ, V, VSP(4), WEIGHT
      COMMON/TOTG/WPLD, D1(15)
C
      A = 1.
      PI = 4. * QATAN(A)
      NOFZ = 'LAUNCH.DAT'
      OPEN (UNIT=10, STATUS='OLD', FILE=NOFZ)
      READ (10, 2500), AINCLN, THETA0, PHI0, TGT, COTG, WPLD,
     +(SDAT(J), J=1, 5), BZ, TYPZ
 2500   FORMAT (1X, F6.2, F7.2, F6.2, 2F5.0, F8.2, 3F9.2, F7.2,
     +F5.1,' ', A1, ' ', A1)
      CLOSE (UNIT=10)
      IF (BZ .EQ. 'y') BZ = 'Y'
      AINCLN = AINCLN * PI / 180.
      A = QCOS(AINCLN) / QCOS(PHI0 * PI / 180.)
      AZH = QASIN(A)
      IF (AINCLN .LE. PI / 2.) HEAD0 = AZH
      IF (AINCLN.GT.PI / 2. .AND. AINCLN.LE.PI) HEAD0=2.*PI + AZH
      IF (AINCLN .GT. PI) HEAD0 = PI + AZH
      IF (TYPZ .EQ. 'D') THEN
```

```
           IF (HEAD0 .LT. PI) THEN
             HEAD0 = HEAD0 + PI
           ELSE
             HEAD0 = HEAD0 - PI
           ENDIF
           SDAT(4) = -SDAT(4)
         ENDIF
         DT0 = 1.
         DT = DT0
         FTPNM = 1852./0.3048
         GAMT = PI / 2.
         G0 = 1.7314E14 / 5710000. ** 2.
         GE = 1.407646882E16 / 2.092567257E7 ** 2.
         HEAD = HEAD0
         IFLAG = 0
         OMEGA = 2.6622E-06
         IORFLAG = 0
         OUTINT = 5.
         OUTTIM  = 0
         PDOT = 0.
         RDOT = .5 * 3.28084
         R0 = 5710000.
         TDOT = OMEGA
         TIM  = 0.
         MU = G0 * R0 ** 2.
         PHI0 = PHI0 * PI / 180.
         THETA0 = THETA0 * PI / 180.
         W(1) = SDAT(1) + WPLD
         M(1) = W(1) / GE
         M(2) = SDAT(2) / GE
         X(8) = M(1)
         X(1) = R0
         X(2) = THETA0
         X(3) = PHI0
         X(4) = M(1)
         IF (TYPZ .EQ. 'A') X(4) = X(4) + M(2)
         X(5) = RDOT
         X(6) = TDOT
         X(7) = PDOT
         RETURN
         END
C         -----------------------------------------------------------
C        |             Integration Subroutine (Runge-Kutta 4)        |
C        |-----------------------------------------------------------|
C        |                                                           |
C        |Name    : RK4                                              |
C        |Athor   : Chris Varner                                     |
```

```
C       |Date     : 3 August, 1986                                              |
C       |                                                                       |
C       |*** Purpose:  This routine is used to update the state                 |
C       |              vector by determining the change in state                |
C       |              during the time step "dt".                               |
C       |                                                                       |
C        ----------------------------------------------------------------
        SUBROUTINE RK4 (DT, GAMP, LNS, OUTTIM, OUTINT, TIM, X)
C       *** Declare Variables **
        IMPLICIT REAL *16 (A-Z)
        INTEGER I, ID1, LNS
C       *** Dimension Arrays ***
        DIMENSION K1(15), K2(15), K3(15), K4(15), RKX(15), RKDX(15)
        DIMENSION X(15)
        COMMON/TOTC/IFLAG, AINCL, AINCLN, ITER, ALAN, ID1, M(4), MU
        COMMON/TOTD/OMEGA, IORFLAG, IOUTFLAG, PEG, PHI0, PI, BZ, R0
        TTEMP = TIM
        DO 10 I=1, 7
          RKX(I) = X(I)
 10     CONTINUE
        CALL EOM (GAMP, RKX, RKDX)
        IF (TIM .GE. OUTTIM) THEN
          CALL OUTPUT (GAMP, LNS, IORFLAG, IOUTFLAG, OUTTIM, OUTINT)
        ENDIF
        DO I= 1, 7
          K1(I) = RKDX(I) * DT
          RKX(I) = X(I) + .5 * K1(I)
        END DO
        TIM  = TIM + 0.5 * DT
        CALL EOM (GAMP, RKX, RKDX)
        DO I=1, 7
          K2(I) = RKDX(I) * DT
          RKX(I) = X(I) + 0.5 * K2(I)
        END DO
        CALL EOM (GAMP, RKX, RKDX)
        DO I=1, 7
          K3(I) = RKDX(I) * DT
          RKX(I) = X(I) + K3(I)
        END DO
        TIM  = TTEMP + DT
        CALL EOM (GAMP, RKX, RKDX)
        DO I=1, 7
          K4(I) = DT * RKDX(I)
          X(I) = X(I)+(K1(I) + 2. * K2(I) + 2. * K3(I) + K4(I)) / 6.
        END DO
        RETURN
        END
```

```
C       ---------------------------------------------------------------
C      |                      Output Subroutine                        |
C      |---------------------------------------------------------------|
C      |                                                               |
C      |Name    : OUTPUT                                               |
C      |Athor   : Chris Varner                                         |
C      |Date    : 12 July, 1988                                        |
C      |                                                               |
C      |*** Purpose:  This routine is used to send all the data to     |
C      |              be displayed to the output device.               |
C      |                                                               |
C       ---------------------------------------------------------------
       SUBROUTINE OUTPUT (GAMP, LNS, IORFLAG, IOUTFLAG, OUTTIM, OUTINT)
C      *** Declare Variables **
       IMPLICIT REAL *16 (A-Z)
       INTEGER ID1,ID2,ID3, I, IFLAG, ITER, J, K, LNS, IORFLAG
       INTEGER IOUTFLAG
       CHARACTER BZ*1, TYPZ*1
C      *** Dimension Arrays ***
       COMMON/TOTA/AOA, ADOT, AOP, APG, COTG, DT, ECC, FTPNM, G0
       COMMON/TOTB/GAMI, GAML, GAMT, GE, H, HEAD0, HEAD, HEADD
       COMMON/TOTC/IFLAG, AINCL, AINCLN, ITER, ALAN, ID1, M(4), MU
       COMMON/TOTD/OMEGA, ID2, ID3, PEG, PHI0, PI, BZ, R0
       COMMON/TOTE/RUNTIM, SDAT(5), T, TGT, THETA0, TIM
       COMMON/TOTF/TRAJDAT(100, 20), TTOW, TYPZ, V, VSP(4), WEIGHT
       COMMON/TOTG/WPLD, X(15)
C
       IF (IOUTFLAG .EQ. 2) THEN
         J = INT(LNS + 1)
         IF (TYPZ .EQ. 'D') THEN
           DO 3510 I=1, (J - 1) / 2
             DO 3500 K=2, 18
               TEMP1 = TRAJDAT(J - I, K)
               TRAJDAT(J - I, K) = TRAJDAT(I, K)
               TRAJDAT(I, K) = TEMP1
 3500        CONTINUE
 3510      CONTINUE
           WRITE (10, 3700), TRAJDAT(J, 10)
           WRITE (10, 3710), TRAJDAT(J, 1), COTG, TRAJDAT(J, 8)
           WRITE (10, 3720), TRAJDAT(J, 9)
           WRITE (10, 3515)
 3515 FORMAT ('       Initial Descent Orbit:'/)
           WRITE (10, 3730), TRAJDAT(J, 1)
           WRITE (10, 3740), TRAJDAT(J, 2)
           TRAJDAT(J, 3) = 180 - TRAJDAT(J, 3)
           WRITE (10, 3750), TRAJDAT(J, 3)
           IF (TRAJDAT(J, 4) .LT. 180.) THEN
```

```
          TRAJDAT(J, 4) = TRAJDAT(J, 4) + 180.
        ELSE
          TRAJDAT(J, 4) = TRAJDAT(J, 4) - 180.
        ENDIF
        WRITE (10, 3760), TRAJDAT(J, 4)
        IF (TRAJDAT(J, 5) .LT. 180.) THEN
          TRAJDAT(J, 5) = 180. - TRAJDAT(J, 5)
        ELSE
          TRAJDAT(J, 5) = 540. - TRAJDAT(J, 5)
        ENDIF
        WRITE (10, 3770), TRAJDAT(J, 5)
        WRITE (10, 3780), TRAJDAT(J, 6)
      ENDIF
      WRITE (10, 3520)
3520 FORMAT (' Time  Altitude  Range  Velocity  Gamma  Heading',
    +'  Thrust  Weight'/' <s>     <ft>     <nm>     <ft/s>',
    +'   <deg>   <deg>   <lbf>   <lbm> '/)
      DO I=1, J-1
        IF (TYPZ .EQ. 'D') THEN
          IF (TRAJDAT(I, 17) .LT. 180.) THEN
            TRAJDAT(I, 17) = TRAJDAT(I, 17) + 180.
          ELSE
            TRAJDAT(I, 17) = TRAJDAT(I, 17) - 180.
          ENDIF
        ENDIF
        WRITE (10,3530), TRAJDAT(I, 1), TRAJDAT(I, 2),
    +TRAJDAT(I, 3), TRAJDAT(I, 6), TRAJDAT(I, 8), TRAJDAT(I, 17),
    +TRAJDAT(I, 11), TRAJDAT(I, 12)
3530 FORMAT (1X,F4.0,F10.0,F7.0,F10.0,F7.2,F8.2,F9.0,F8.0)
      END DO
      WRITE (10, 3550), TRAJDAT(J, 7)
3550    FORMAT (/' Ideal Performance Delta Velocity is: ',F8.2,
    +' <ft/s>'/)
      IF (TYPZ .EQ. 'A') THEN
        WRITE (10, 3555)
3555      FORMAT (/'     Boost Orbit:'/)
        WRITE (10, 3730), TRAJDAT(J, 1)
        WRITE (10, 3740), TRAJDAT(J, 2)
        WRITE (10, 3750), TRAJDAT(J, 3)
        WRITE (10, 3760), TRAJDAT(J, 4)
        WRITE (10, 3770), TRAJDAT(J, 5)
        WRITE (10, 3780), TRAJDAT(J, 6)
        WRITE (10, 3560), TRAJDAT(J, 1), COTG, TRAJDAT(J, 9)
3560 FORMAT (/'   Velocity Required at Apocynthion to Achieve'/
    +'   the Holding Orbit ( ',F4.0,' X ',F4.0,' )    :',
    +F9.2,' <ft/s>'/)
        TRAJDAT(J, 10) = -TRAJDAT(J, 10)
```

```
         WRITE (10, 3570), TRAJDAT(J, 10)
3570 FORMAT ('   Fuel Required for the Apocynthion Burn   :',F9.2,
    +' <lbm>'/)
         WRITE (10, 3580), TRAJDAT(J, 11)
3580 FORMAT ('   Weight After Apocynthion Burn            :',F9.2,
    +' <lbm>'/)
         WRITE (10, 3590), TRAJDAT(J, 8)
3590 FORMAT ('   Weight of the Payload Placed in Orbit:',F9.2,
    +' <lbm>')
       ENDIF
       WRITE (10, 3600)
3600 FORMAT (/' ************ SIMULATION COMPLETE **************')
       IOUTFLAG = 0
     ELSE
       IF (IOUTFLAG .EQ. 1) THEN
         ITER = ITER + 1
         WRITE (5, 3610), ITER, APG, PEG
3610 FORMAT (' Iteration # ',I3,'     Apocynthion = ',F7.1,' <nm>',
    +      '       Pericynthion = ',F7.1,' <nm>')
         IOUTFLAG = 0
       ELSE
         IF (IORFLAG .EQ. 1) THEN
           AINCLD = AINCL * 180. / PI
           WRITE (10, 3620), APG, PEG, AINCLD, ALAN, AOP, ECC
3620         FORMAT (1X, F7.1, F7.1, F8.2, F8.2, F8.2, F7.4)
           IF (TYPZ .EQ. 'D') THEN
             DV = -SDAT(4) * GE * LOG(X(4) / M(1))
           ELSE
             DV = SDAT(4) * GE * LOG((M(1) + M(2)) / X(4))
           ENDIF
           WRITE (10, 3630), DV
3630         FORMAT (1X,F6.1)
     TEMP = 2./(R0+APG*FTPNM)-2./(2.*R0+(APG+COTG)*FTPNM)
           DV2 = QSQRT(MU * TEMP)
     TEMP = 2./(R0+APG*FTPNM)-2./(2.*R0+(APG+PEG)*FTPNM)
           DV2 = DV2 - QSQRT(MU * TEMP)
           MF = X(4) * EXP(-DV2 / SDAT(4) / GE)
           MFUEL = MF - X(4)
           IF (TYPZ .EQ. 'A') THEN
             WPLD = MF * GE - SDAT(1)
             WRITE (10, 3640), WPLD
3640           FORMAT (1X,F6.0)
           ENDIF
           WFUEL = MFUEL * GE
           WF = MF * GE
           WRITE (10, 3650), DV2, WFUEL, WF
3650         FORMAT (1X,F8.1, F8.2, F9.0)
```

```
            IORFLAG = 0
          ELSE
            TPAD = THETA0 + OMEGA * TIM
            PPAD = PHI0
            RANGE2 = R0 / FTPNM * (X(2) - TPAD)
            RANGE3 = R0 / FTPNM * (X(3) - PPAD)
            A = QCOS(X(2) - TPAD) * QCOS(X(3) - PPAD)
            RANGE = QACOS(A) * R0 / FTPNM
            ALONGD = (X(2) - OMEGA * TIM ) * 180. / PI
            ALATD = X(3) * 180. / PI
            GAMLD = GAML*180./PI
            GAMID = GAMI*180./PI
            LNS = LNS + 1
            IF (BZ .EQ. 'Y') THEN
              WRITE (5, 3660), TIM , H, V, GAMLD, T,
     +WEIGHT
3660          FORMAT (1X, F5.0, F9.0, F7.0, F6.2, 2F9.0)
            ENDIF
            WRITE (10, 3670), TIM , H, RANGE, ALONGD, ALATD, V,
     +VSP(4), GAMLD, GAMID, AOA, T, WEIGHT,
     +ADOT, RANGE2, RANGE3, TTOW, HEADD
3670 FORMAT (1X, F5.0, F8.0, F7.0, 2F7.2, 2F7.0, 3F6.2, 2F9.0,
     +F6.2, 2F7.0, F5.2, F6.2)
            DO WHILE (OUTTIM .LE. TIM)
              OUTTIM  = OUTTIM  + OUTINT
              OUTTIM  = QFLOAT(INT(OUTTIM ))
            END DO
          ENDIF
        ENDIF
      ENDIF
3700 FORMAT (/'      Weight Prior to Deorbit Burn      :',F9.2,
     +' <lbm>'/)
3710 FORMAT ('     Delta Velocity Required to Deorbit'/'     ',
     +'from the Holding Orbit ( ',F4.0,' X ',F4.0,' )'/'     to',
     +' the Initial Descent Orbit      :',F9.2,' <ft/s>'/)
3720 FORMAT ('     Weight of Fuel Required to Deorbit:',F9.2,
     +' <lbm>'//)
3730 FORMAT ('     Apocynthion                        -- ', F9.4,
     +' <nm>')
3740 FORMAT ('     Pericynthion                       -- ', F9.4,
     +' <nm>')
3750 FORMAT ('     Inclination                        -- ', F9.4,
     +' <deg>')
3760 FORMAT ('     Longitude of the Ascending Node -- ', F9.4,
     +' <deg>')
3770 FORMAT ('     Argument of Pericynthion           -- ', F9.4,
     +' <deg>')
```

```
 3780 FORMAT ('      Eccentricity                          -- ', F9.4,
     +' <nd>'/)
      RETURN
      END
C      ------------------------------------------------------------
C     |                 Orbit Calculation Subroutine                |
C     |-------------------------------------------------------------|
C     |                                                             |
C     |Name    : ORBIT                                              |
C     |Athor   : Chris Varner                                       |
C     |Date    : 18 March, 1986                                     |
C     |                                                             |
C     |*** Purpose: This routine calculates the orbital parameters|
C     |              based on the position and velocity of the      |
C     |              spacecraft with respect to the planet about    |
C     |              which the orbit is to be determined.           |
C     |                                                             |
C      ------------------------------------------------------------
      SUBROUTINE ORBIT (VSP,X,AOP,APG,ECC,AINCL,ALAN,PEG,SMJ)
C     *** Declare Variables **
      IMPLICIT REAL *16 (A-Z)
      INTEGER I, IFLAG, ITER, J, K, LNS, IORFLAG, IOUTFLAG
      CHARACTER BZ*1, TYPZ*1
C     *** Dimension Arrays ***
      DIMENSION C(3, 3), PSN(4), VEL(4), VSP(4), X(15)
      COMMON/TOTA/AOA, ADOT, D1, D2, COTG, DT, D3, FTPNM, G0
      COMMON/TOTB/GAMI, GAML, GAMT, GE, H, HEAD0, HEAD, HEADD
      COMMON/TOTC/IFLAG, D4, AINCLN, ITER, D5, LNS, M(4), MU
      COMMON/TOTD/OMEGA, IORFLAG, IOUTFLAG, D55, PHI0, PI, BZ, R0
      COMMON/TOTE/RUNTIM, SDAT(5), T, TGT, THETA0, TIM
      COMMON/TOTF/TRAJDAT(100, 20), TTOW, TYPZ, V, D6(4), WEIGHT
      COMMON/TOTG/WPLD, D7(15)
C
      C(1, 1) = QCOS(X(3)) * QCOS(X(2))
      C(1, 2) = QCOS(X(3)) * QSIN(X(2))
      C(1, 3) = QSIN(X(3))
      C(2, 1) = -QSIN(X(2))
      C(2, 2) = QCOS(X(2))
      C(2, 3) = 0.
      C(3, 1) = -QSIN(X(3)) * QCOS(X(2))
      C(3, 2) = -QSIN(X(3)) * QSIN(X(2))
      C(3, 3) = QCOS(X(3))
      DO I=1, 3
        TEMP = 0
        DO J=1, 3
          TEMP = TEMP + VSP(J) * C(J, I)
        END DO
```

```
      VEL(I) = TEMP
    END DO
    PSN(1) = X(1) * QCOS(X(3)) * QCOS(X(2))
    PSN(2) = X(1) * QCOS(X(3)) * QSIN(X(2))
    PSN(3) = X(1) * QSIN(X(3))
    PSN(4) = QSQRT(PSN(1) ** 2. + PSN(2) ** 2. + PSN(3) ** 2.)
    VEL(4) = QSQRT(VEL(1) ** 2. + VEL(2) ** 2. + VEL(3) ** 2.)
    SMJ = 1. / (2. / PSN(4) - VEL(4) ** 2. / MU)
    AI = PSN(1) * VEL(1) + PSN(2) * VEL(2) + PSN(3) * VEL(3)
    IF (SMJ .LT. 0) THEN
      PRINT *, '***** Hyperbolic Orbit *****'
      PRINT *, 'Try again with a shorter MECO time'
      STOP
    ENDIF
    DENOM = 1. - PSN(4) / SMJ
    NUMOR = AI / QSQRT(MU * SMJ)
    ECC = QSQRT(DENOM ** 2. + NUMOR ** 2.)
    ECA = QATAN2(NUMOR, DENOM)
    MEA = ECA - ECC * QSIN(ECA)
    AK = MU * DENOM / PSN(4)
    SLR = SMJ * (1. - ECC ** 2.)
    AJ = PSN(4) - SLR
    AL = AI / PSN(4)
    DO I=1, 3
      C(1, I) = (AK * PSN(I) - AI * VEL(I)) / MU / ECC
      C(2, I) = (AL * PSN(I)-AJ * VEL(I)) / ECC / QSQRT(MU * SLR)
    END DO
    C(3, 1) = C(1, 2) * C(2, 3) - C(1, 3) * C(2, 2)
    C(3, 2) = C(1, 3) * C(2, 1) - C(1, 1) * C(2, 3)
    C(3, 3) = C(1, 1) * C(2, 2) - C(1, 2) * C(2, 1)
    IF (C(3, 3) .GT. 1.) C(3, 3) = 1.
    AINCL = QACOS(C(3, 3))
    ALAN = QATAN2(C(3, 1), -C(3, 2)) * 180. / PI
    IF (ALAN .LT. 0) ALAN = 360 + ALAN
    AOP = QATAN2(C(1, 3), C(2, 3)) * 180. / PI
    IF (AOP .LT. 0) AOP = 360 + AOP
    APG = (SMJ * (1 + ECC) - R0) / FTPNM
    PEG = (SMJ * (1 - ECC) - R0) / FTPNM
    RETURN
    END
C    ------------------------------------------------------------
C    |                Equations of Motion Subroutine            |
C    |----------------------------------------------------------|
C    |                                                          |
C    |Name   : EOM                                              |
C    |Athor  : Chris Varner                                     |
C    |Date   : 25 June, 1988                                    |
```

```
C      |                                                             |
C      |*** Purpose: This routine is used to evaluate the equations|
C      |             motion.                                         |
C      |                                                             |
C       -------------------------------------------------------------
       SUBROUTINE EOM (GAMP, RKX, RKDX)
C      *** Declare Variables **
       IMPLICIT REAL *16 (A-Z)
       INTEGER I, IFLAG, ITER, J, K, LNS, IORFLAG, IOUTFLAG
       CHARACTER BZ*1, TYPZ*1
C      *** Dimension Arrays ***
       DIMENSION RKX(15), RKDX(15)
       COMMON/TOTA/AOA, ADOT, AOP, APG, COTG, DT, ECC, FTPNM, G0
       COMMON/TOTB/GAMI, GAML, GAMT, GE, H, HEAD0, HEAD, HEADD
       COMMON/TOTC/IFLAG, AINCL, AINCLN, ITER, ALAN, LNS, M(4), MU
       COMMON/TOTD/OMEGA, IORFLAG, IOUTFLAG, PEG, PHI0, PI, BZ, R0
       COMMON/TOTE/RUNTIM, SDAT(5), T, TGT, THETA0, TIM
       COMMON/TOTF/TRAJDAT(100, 20), TTOW, TYPZ, V, VSP(4), WEIGHT
       COMMON/TOTG/WPLD, X(15)
C
C      *** Preliminary Calculations ***
       RKX(8) = RKX(4)
       VSP(1) = RKX(5)
       VSP(2) = RKX(1) * RKX(6) * QCOS(RKX(3))
       VSP(3) = RKX(1) * RKX(7)
       VSP(4) = QSQRT(VSP(1) ** 2. + VSP(2) ** 2. + VSP(3) ** 2.)
       RDA = R0 * OMEGA * QCOS(PHI0)
       V2 = VSP(2) - RDA
       V = QSQRT(VSP(1) ** 2. + V2 ** 2. + VSP(3) ** 2.)
       H = RKX(1) - R0
       GAMI = QATAN(VSP(1) /  QSQRT(VSP(2) ** 2. + VSP(3) ** 2.))
       IF (V2 .EQ. 0. .AND. VSP(3) .EQ. 0.) THEN
         GAML = 90. * PI / 180.
       ELSE
         GAML = QATAN(VSP(1) / QSQRT(V2 ** 2. + VSP(3) ** 2.))
       ENDIF
       NUMOR = RKX(7)
       DENOM = RKX(6) - OMEGA
       IF (DENOM .EQ. 0.) THEN
         IF (NUMOR .GE. 0.) THEN
           ANGLE = PI / 2.
         ELSE
           ANGLE = -PI / 2.
         ENDIF
       ELSE
         ANGLE = QATAN2(NUMOR, DENOM)
       ENDIF
```

```
      IF (ANGLE .LE. PI / 2.) THEN HEAD = PI / 2. - ANGLE
      IF (ANGLE .GT. PI / 2.) THEN HEAD = 5 * PI / 2. - ANGLE
      CALL CONTROL (GAML, GAMP, V, GAMT, HEAD, HEADT)
      CALL PROFILE (GAMP, RKX, LEVEL)
      T = SDAT(3) * LEVEL
      MDOT = -T / GE / SDAT(4)
      T = T
      HEADD = HEAD * 180. / PI
      WEIGHT = RKX(4) * GE
      TTOW = T / WEIGHT
      DO I=1, 7
        RKX(I + 8) = RKX(I)
      END DO
C     ******************************************************
C     *** Equations of Motion for Spherical Coordinates ***
C     ******************************************************
      RKDX(1) = RKX(5)
      RKDX(2) = RKX(6)
      RKDX(3) = RKX(7)
      RKDX(4) = MDOT
      TEMP1 = 0
      TEMP2 = 0
      TEMP3 = 0
      TEMP1 = T * QSIN(GAMT)
      TEMP1 = TEMP1 / RKX(4)
      TEMP2=(T*QCOS(GAMT)*QSIN(HEADT))/(RKX(4)*RKX(1)*QCOS(RKX(3)))
      TEMP3 = (T * QCOS(GAMT) * QCOS(HEADT)) / (RKX(4) * RKX(1))
      RKDX(5)=RKX(1)*RKX(6)**2.*QCOS(RKX(3))**2.+RKX(1)*RKX(7)**2.
      RKDX(5) = RKDX(5) - MU / RKX(1) ** 2. + TEMP1
      RKDX(6)=2.*(-RKX(5)*RKX(6)/RKX(1)+RKX(6)*RKX(7)*TAN(RKX(3)))
      RKDX(6) = RKDX(6) + TEMP2
      RKDX(7) = -2. * RKX(5) * RKX(7) / RKX(1)
      RKDX(7)=RKDX(7)-RKX(6)**2.*QSIN(RKX(3))*QCOS(RKX(3))+TEMP3
      RETURN
      END
C      ----------------------------------------------------------
C     |                  Thrust Profile Subroutine              |
C     |----------------------------------------------------------|
C     |                                                          |
C     |Name    : THRUST                                          |
C     |Athor   : Chris Varner                                    |
C     |Date    : 3 July, 1988                                    |
C     |                                                          |
C     |*** Purpose:  The Thrust Profile subroutine provides the  |
C     |              equations of motion with the level of thrust|
C     |              supplied by the engines.                    |
C     |                                                          |
```

```
C     ------------------------------------------------------------
      SUBROUTINE PROFILE (GAMP, RKX, LEVEL)
C     *** Declare Variables **
      IMPLICIT REAL *16 (A-Z)
      INTEGER IFLAG, ITER, LNS, IORFLAG, IOUTFLAG
      CHARACTER BZ*1, TYPZ*1
C     *** Dimension Arrays ***
      DIMENSION RKX(15)
      COMMON/TOTA/AOA, ADOT, AOP, APG, COTG, DT, ECC, FTPNM, G0
      COMMON/TOTB/GAMI, GAML, GAMT, GE, H, HEAD0, HEAD, HEADD
      COMMON/TOTC/IFLAG, AINCL, AINCLN, ITER, ALAN, LNS, M(4), MU
      COMMON/TOTD/OMEGA, IORFLAG, IOUTFLAG, PEG, PHI0, PI, BZ, R0
      COMMON/TOTE/RUNTIM, SDAT(5), T, TGT, THETA0, TIM
      COMMON/TOTF/TRAJDAT(100, 20), TTOW, TYPZ, V, VSP(4), WEIGHT
      COMMON/TOTG/WPLD, X(15)
C
      IF (TYPZ .EQ. 'D') THEN
        IF (TIM  .GT. SDAT(5)) THEN
          IF (TIM .LE. SDAT(5) + 35.) THEN
            LEVEL = PRF1 + (1. - PRF1) / 35. * (TIM - SDAT(5))
          ELSE
            LEVEL = 1.
          ENDIF
        ELSE
          LEVEL = RKX(4) * G0 / SDAT(3)
          PRF1 = LEVEL
        ENDIF
      ELSE
        LEVEL = 1.
        PRF1 = 1.
      ENDIF
      RETURN
      END
C     ------------------------------------------------------------
C     |                  Control Procedures Subroutine              |
C     |-----------------------------------------------------------|
C     |                                                           |
C     |Name    : CONTROL                                          |
C     |Athor   : Chris Varner                                     |
C     |Date    : 15 June, 1988                                    |
C     |                                                           |
C     |*** Purpose: This subroutine supplies the thrust control   |
C     |             procedures required for the ascent and descent|
C     |             launches and landings.                        |
C     |                                                           |
C     ------------------------------------------------------------
      SUBROUTINE CONTROL (GAML, GAMP, V, GAMT, HEAD, HEADT)
```

```
C       *** Declare Variables **
        IMPLICIT REAL *16 (A-Z)
        INTEGER IFLAG, ITER, LNS, IORFLAG, IOUTFLAG, IPFLAG
        CHARACTER BZ*1, TYPZ*1
        COMMON/TOTA/AOA, ADOT, AOP, APG, COTG, DT, ECC, FTPNM, G0
        COMMON/TOTB/GAMI, D1, D2, GE, H, HEAD0, D3, HEADD
        COMMON/TOTC/IFLAG, AINCL, AINCLN, ITER, ALAN, LNS, M(4), MU
        COMMON/TOTD/OMEGA, IORFLAG, IOUTFLAG, PEG, PHI0, PI, BZ, R0
        COMMON/TOTE/RUNTIM, SDAT(5), T, TGT, THETA0, TIM
        COMMON/TOTF/TRAJDAT(100, 20), TTOW, TYPZ, D4, VSP(4), WEIGHT
        COMMON/TOTG/WPLD, X(15)
C
        IF (V .LT. 30.) THEN
          GAMT = PI / 2.
          HEADT = HEAD0
          IPFLAG = 0
        ELSE
          IF (IPFLAG .LT. 20) THEN
            IF (IPFLAG .LE. 10) THEN
              GAMT = GAMP - (90.*PI/180. - GAMP)/10.*QFLOAT(IPFLAG)
              HEADT = HEAD0
              IPFLAG = IPFLAG + 1
            ELSE
              GAMT = GAMP + (GAML - GAMP)/10.*QFLOAT(IPFLAG - 10)
              IPFLAG = IPFLAG + 1
            ENDIF
          ELSE
            GAMT = GAML
            IF (GAMT .LT. 0.) THEN GAMT = 0
            IF (GAML .GT. 80.* PI/180.) THEN
              HEADT = HEAD0
              HEAD = HEAD0
            ELSE
              HEADT = HEAD0
            ENDIF
          ENDIF
        ENDIF
        RETURN
        END
```

APPENDIX C: Input/Output Examples

The following examples are simulations of the Apollo 15 descent to and ascent from the lunar surface. The spacecraft characteristics and the trajectory data are taken from the Apollo 15 Mission Report, document MSC-05161, written by the National Aeronautics and Space Administration's Manned Spacecraft Center (Houston, Texas) in December of 1971. All of the pertinent data is condensed into Table A3.

Table A3: Apollo 15 Mission Characteristics

Holding Orbit 50.1 <nm> X 56.4 <nm>

Orbit Inclination 26.2°

Landing Site Location Hadley Rille-Apennine Mountains Region
26.1011° North Latitude
3.6528° East Longitude

Mass of Lunar Module at:

Separation	35,718 <lb>
Lunar Landing	18,175 <lb>
Lunar Lift-off	10,915 <lb>
Docking	5,826 <lb>

Engine Performance:

	Thrust	Specific Impulse
Descent Engine	9,750 <lbf>	303 <s>
Ascent Engine	3,500 <lbf>	306 <s>

78 seconds elapsed between commencing Attitude Hold and Touchdown on the lunar surface. It is estimated that approximately 60 seconds were spent hovering.

The results of these simulations compare favorably with those of the actual Apollo 15 flight data shown in Table A3. The descent simulation predicts a weight prior to the deorbit burn of 35,642 lb; the actual value for Apollo 15 was 35,718 lb. The ascent simulation is of equivalent accuracy. Predicting a post apocynthion burn weight of 5,754 lb, the simulation is only 72 lb less than the actual weight recorded in the Mission Report.

LANDER Descent Simulation of the Apollo 15 Lunar Descent Module

The following inputs are supplied at the program prompts:

IS THIS TO BE AN ASCENT OR DESCENT SIMULATION ?
Answer: D

LANDING SITE LATITUDE (-90 TO +90)
Answer: 26.1011

LANDING SITE LONGITUDE (0 TO 360)
Answer: 3.6527

INERT WEIGHT <LB>
Answer: 18175

PROPELLANT WEIGHT <LB>
Answer: 17543

THRUST <LBF>
Answer: 9750

SPECIFIC IMPULSE <S>
Answer: 303

HOVER TIME <S>
Answer: 60

PAYLOAD WEIGHT <LB>
Answer: 0

TIME TO MAIN ENGINE CUT-OFF (MECO)? <S>
Answer: 440

HOLDING ORBIT PERICYNTHION <NM>
Answer: 50

HOLDING ORBIT APOCYNTHION <NM>
Answer: 50

FLIGHT PATH ANGLE AT PITCH-OVER ?
Answer: 70

HOLDING ORBIT INCLINATION ? (0 TO 360)
Answer: 26.2

DO YOU WISH TO SEE THE TRAJECTORY OF EACH ITERATION ?
Answer: N

OUTPUT:

```
Weight Prior to Deorbit Burn      : 35642.00 <lbm>

Delta Velocity Required to Deorbit
from the Holding Orbit (  48. X  50. )
to the Initial Descent Orbit      :    34.90 <ft/s>

Weight of Fuel Required to Deorbit:   127.47 <lbm>


 Initial Descent Orbit:

Apocynthion                        --    48.5000 <nm>
Pericynthion                       --    24.7000 <nm>
Inclination                        --    25.8400 <deg>
Longitude of the Ascending Node    --   274.2300 <deg>
Argument of Pericynthion           --    75.1600 <deg>
Eccentricity                       --     0.0122 <nd>
```

Time <s>	Altitude <ft>	Range <nm>	Velocity <ft/s>	Gamma <deg>	Heading <deg>	Thrust <lbf>	Weight <lbm>
0.	149917.	258.	5465.	0.00	87.63	9750.	35450.
5.	149918.	253.	5421.	0.00	87.63	9750.	35289.
10.	149918.	248.	5376.	0.00	87.63	9750.	35128.
15.	149914.	243.	5332.	0.01	87.63	9750.	34967.
20.	149905.	239.	5287.	0.03	87.63	9750.	34806.
25.	149889.	234.	5242.	0.04	87.63	9750.	34645.
30.	149864.	229.	5197.	0.07	87.63	9750.	34485.
35.	149827.	225.	5151.	0.10	87.63	9750.	34324.
40.	149777.	220.	5106.	0.13	87.63	9750.	34163.
45.	149712.	216.	5060.	0.16	87.63	9750.	34002.
50.	149631.	211.	5014.	0.21	87.63	9750.	33841.
55.	149531.	207.	4968.	0.25	87.63	9750.	33680.
60.	149411.	203.	4921.	0.31	87.63	9750.	33519.
65.	149268.	198.	4875.	0.36	87.63	9750.	33358.
70.	149102.	194.	4828.	0.42	87.63	9750.	33197.
75.	148911.	190.	4781.	0.49	87.63	9750.	33037.
80.	148692.	186.	4733.	0.56	87.63	9750.	32876.
85.	148445.	181.	4686.	0.64	87.63	9750.	32715.
90.	148167.	177.	4638.	0.72	87.63	9750.	32554.

95.	147857.	173.	4591.	0.81	87.63	9750.	32393.
100.	147515.	169.	4542.	0.91	87.63	9750.	32232.
105.	147137.	165.	4494.	1.01	87.63	9750.	32071.
110.	146723.	161.	4446.	1.11	87.63	9750.	31910.
115.	146272.	157.	4397.	1.23	87.63	9750.	31749.
120.	145781.	153.	4348.	1.34	87.63	9750.	31588.
125.	145251.	149.	4299.	1.47	87.63	9750.	31428.
130.	144679.	146.	4250.	1.60	87.63	9750.	31267.
135.	144064.	142.	4200.	1.74	87.63	9750.	31106.
140.	143405.	138.	4151.	1.88	87.63	9750.	30945.
145.	142702.	134.	4101.	2.03	87.63	9750.	30784.
150.	141952.	131.	4051.	2.19	87.63	9750.	30623.
155.	141155.	127.	4001.	2.35	87.63	9750.	30462.
160.	140310.	124.	3950.	2.52	87.63	9750.	30301.
165.	139416.	120.	3899.	2.70	87.63	9750.	30140.
170.	138473.	117.	3849.	2.89	87.63	9750.	29980.
175.	137478.	113.	3798.	3.08	87.63	9750.	29819.
180.	136432.	110.	3746.	3.28	87.63	9750.	29658.
185.	135334.	107.	3695.	3.49	87.63	9750.	29497.
190.	134184.	103.	3643.	3.71	87.63	9750.	29336.
195.	132980.	100.	3591.	3.93	87.63	9750.	29175.
200.	131722.	97.	3539.	4.16	87.63	9750.	29014.
205.	130409.	94.	3487.	4.41	87.63	9750.	28853.
210.	129043.	91.	3435.	4.66	87.63	9750.	28692.
215.	127621.	88.	3382.	4.92	87.63	9750.	28532.
220.	126143.	85.	3329.	5.19	87.63	9750.	28371.
225.	124611.	82.	3276.	5.47	87.63	9750.	28210.
230.	123023.	79.	3223.	5.75	87.63	9750.	28049.
235.	121379.	76.	3170.	6.05	87.63	9750.	27888.
240.	119680.	73.	3116.	6.36	87.63	9750.	27727.
245.	117926.	71.	3062.	6.68	87.63	9750.	27566.
250.	116117.	68.	3008.	7.01	87.63	9750.	27405.
255.	114252.	65.	2954.	7.36	87.63	9750.	27244.
260.	112334.	63.	2900.	7.71	87.63	9750.	27084.
265.	110362.	60.	2845.	8.08	87.63	9750.	26923.
270.	108337.	58.	2791.	8.45	87.63	9750.	26762.
275.	106260.	55.	2736.	8.85	87.63	9750.	26601.
280.	104130.	53.	2681.	9.25	87.63	9750.	26440.
285.	101951.	50.	2625.	9.67	87.63	9750.	26279.
290.	99722.	48.	2570.	10.10	87.63	9750.	26118.
295.	97444.	46.	2514.	10.55	87.63	9750.	25957.
300.	95119.	44.	2459.	11.01	87.63	9750.	25796.
305.	92749.	41.	2403.	11.49	87.63	9750.	25636.
310.	90335.	39.	2347.	11.98	87.63	9750.	25475.
315.	87878.	37.	2290.	12.49	87.63	9750.	25314.
320.	85381.	35.	2234.	13.02	87.63	9750.	25153.
325.	82845.	33.	2177.	13.57	87.63	9750.	24992.

330.	80273.	32.	2121.	14.13	87.63	9750.	24831.
335.	77667.	30.	2064.	14.72	87.63	9750.	24670.
340.	75030.	28.	2007.	15.33	87.63	9750.	24509.
345.	72363.	26.	1950.	15.96	87.63	9750.	24348.
350.	69671.	25.	1892.	16.61	87.63	9750.	24187.
355.	66956.	23.	1835.	17.28	87.63	9750.	24027.
360.	64221.	21.	1777.	17.98	87.63	9750.	23866.
365.	61469.	20.	1720.	18.71	87.63	9750.	23705.
370.	58705.	18.	1662.	19.47	87.63	9750.	23544.
375.	55933.	17.	1604.	20.25	87.63	9750.	23383.
380.	53156.	16.	1546.	21.06	87.63	9750.	23222.
385.	50378.	14.	1488.	21.91	87.63	9750.	23061.
390.	47606.	13.	1429.	22.80	87.63	9750.	22900.
395.	44842.	12.	1371.	23.71	87.63	9750.	22739.
400.	42093.	11.	1313.	24.67	87.63	9750.	22579.
405.	39365.	10.	1254.	25.68	87.63	9750.	22418.
410.	36662.	9.	1195.	26.72	87.63	9750.	22257.
415.	33991.	8.	1137.	27.82	87.63	9750.	22096.
420.	31359.	7.	1078.	28.96	87.63	9750.	21935.
425.	28773.	6.	1019.	30.17	87.63	9750.	21774.
430.	26239.	5.	961.	31.43	87.63	9750.	21613.
435.	23765.	5.	902.	32.77	87.63	9750.	21452.
440.	21360.	4.	843.	34.18	87.63	9750.	21291.
445.	19033.	3.	784.	35.67	87.63	9750.	21131.
450.	16791.	3.	725.	37.25	87.63	9750.	20970.
455.	14645.	2.	667.	38.94	87.63	9750.	20809.
460.	12604.	2.	608.	40.74	87.63	9750.	20648.
465.	10680.	2.	549.	42.67	87.63	9750.	20487.
470.	8884.	1.	491.	44.76	87.63	9750.	20326.
475.	7227.	1.	433.	47.04	87.63	9750.	20165.
480.	5723.	1.	374.	49.53	87.63	9750.	20004.
485.	4384.	0.	316.	52.30	87.63	9750.	19843.
490.	3226.	0.	258.	55.42	87.63	9750.	19683.
495.	2263.	0.	201.	59.02	87.63	9750.	19522.
500.	1505.	0.	147.	63.28	87.63	8800.	19369.
505.	941.	0.	102.	68.34	87.63	7851.	19231.
510.	549.	0.	65.	74.26	87.63	6901.	19109.
515.	303.	0.	37.	83.35	87.63	5951.	19003.
520.	169.	0.	18.	90.00	87.63	5002.	18913.
525.	114.	0.	6.	89.99	87.63	4052.	18838.
530.	99.	0.	2.	89.97	87.63	3102.	18779.
535.	90.	0.	2.	89.97	87.63	3094.	18728.
540.	82.	0.	2.	89.97	87.63	3085.	18677.
545.	74.	0.	2.	89.97	87.63	3077.	18626.
550.	66.	0.	2.	89.98	87.63	3069.	18576.
555.	57.	0.	2.	89.98	87.63	3060.	18525.
560.	49.	0.	2.	89.98	87.63	3052.	18475.

565.	41.	0.	2.	89.99	87.63	3044.	18424.
570.	33.	0.	2.	89.99	87.63	3035.	18374.
575.	25.	0.	2.	89.99	87.63	3027.	18324.
580.	16.	0.	2.	89.99	87.63	3019.	18274.
585.	8.	0.	2.	90.00	87.63	3011.	18225.
590.	0.	0.	2.	90.00	87.63	3002.	18175.

Ideal Performance Delta Velocity is: 6525.00 <ft/s>

************ SIMULATION COMPLETE ***************

<u>LANDER Ascent Simulation of the Apollo 15 Lunar Ascent Module</u>

The following inputs are supplied at the program prompts:

IS THIS TO BE AN ASCENT OR DESCENT SIMULATION ?
Answer: A

LANDING SITE LATITUDE (-90 TO +90)
Answer: 26.1011

LANDING SITE LONGITUDE (0 TO 360)
Answer: 3.6527

INERT WEIGHT <LB>
Answer: 5326 [1]

PROPELLANT WEIGHT <LB>
Answer: 5589*

THRUST <LBF>
Answer: 3500

SPECIFIC IMPULSE <S>
Answer: 306

HOVER TIME <S>
Answer: (Not Applicable)

PAYLOAD WEIGHT <LB>
Answer: 0

TIME TO MAIN ENGINE CUT-OFF (MECO)? <S>
Answer: 460

HOLDING ORBIT PERICYNTHION <NM>
Answer: 50

---

[1] 500 pounds is transfered to the propellant* from the inert to prevent the simulation vehicle from running out of propellant during the simulation. This has no effect upon the results because the only propellant used is that necessary to achieve the requested orbit; the remaining fuel is assumed to be payload or otherwise inert. If the propellant were to be exhausted during the ascent then the iteration technique may become divergent, and fail to reach a satisfactory solution.

HOLDING ORBIT APOCYNTHION <NM>

Answer: 50

FLIGHT PATH ANGLE AT PITCH-OVER ?

Answer: 85 [2]

HOLDING ORBIT INCLINATION ? (0 TO 360)

Answer: 26.2

DO YOU WISH TO SEE THE TRAJECTORY OF EACH ITERATION ?

Answer: N

OUTPUT:

Time <s>	Altitude <ft>	Range <nm>	Velocity <ft/s>	Gamma <deg>	Heading <deg>	Thrust <lbf>	Weight <lbm>
0.	0.	0.	2.	90.00	87.63	3500.	10915.
5.	71.	0.	27.	90.00	87.63	3500.	10858.
10.	267.	0.	52.	84.34	87.63	3500.	10801.
15.	590.	0.	78.	81.47	87.63	3500.	10743.
20.	1038.	0.	104.	78.58	87.63	3500.	10686.
25.	1612.	0.	131.	75.71	87.63	3500.	10629.
30.	2310.	0.	159.	72.87	87.63	3500.	10572.
35.	3129.	0.	187.	70.08	87.63	3500.	10515.
40.	4067.	0.	216.	67.35	87.63	3500.	10457.
45.	5120.	0.	246.	64.68	87.63	3500.	10400.
50.	6286.	0.	276.	62.07	87.63	3500.	10343.
55.	7559.	1.	308.	59.55	87.63	3500.	10286.
60.	8936.	1.	340.	57.09	87.63	3500.	10229.
65.	10412.	1.	373.	54.72	87.63	3500.	10172.
70.	11981.	1.	407.	52.42	87.63	3500.	10114.
75.	13639.	1.	443.	50.21	87.63	3500.	10057.
80.	15380.	2.	479.	48.08	87.63	3500.	10000.
85.	17199.	2.	516.	46.02	87.63	3500.	9943.

---

[2] In this case, the recommended initial guess of 70° is bad. Due to the slow ascent, the spacecraft is very sensitive to the pitch-over angle. Any choice of angle less than 80° will result in the vehicle pitching-over too rapidly -- flying back into the ground. The result is that the program falls into what is often referred to as "Bang-Bang" instability, where it continuously oscillates between two, three, or four pitch-over angles. This endless loop can be spotted by noting that the periodic screen output is cycling between the same set of apocynthion and pericynthion. The problem can be determined by printing the trajectory of each iteration. Once the problem has been analyzed, then modifications of the input data can be implemented. In this case the solution was to raise the pitch-over angle.

90.	19089.	2.	554.	44.04	87.63	3500.	9886.
95.	21047.	3.	593.	42.14	87.63	3500.	9828.
100.	23066.	3.	633.	40.31	87.63	3500.	9771.
105.	25140.	4.	674.	38.55	87.63	3500.	9714.
110.	27264.	4.	716.	36.86	87.63	3500.	9657.
115.	29434.	5.	759.	35.24	87.63	3500.	9600.
120.	31642.	5.	803.	33.69	87.63	3500.	9542.
125.	33885.	6.	848.	32.19	87.63	3500.	9485.
130.	36157.	6.	894.	30.76	87.63	3500.	9428.
135.	38453.	7.	940.	29.38	87.63	3500.	9371.
140.	40768.	8.	988.	28.06	87.63	3500.	9314.
145.	43098.	9.	1036.	26.79	87.63	3500.	9257.
150.	45439.	10.	1086.	25.57	87.63	3500.	9199.
155.	47784.	11.	1136.	24.40	87.63	3500.	9142.
160.	50131.	12.	1187.	23.28	87.63	3500.	9085.
165.	52475.	13.	1239.	22.20	87.63	3500.	9028.
170.	54813.	14.	1292.	21.16	87.63	3500.	8971.
175.	57140.	15.	1346.	20.17	87.63	3500.	8913.
180.	59452.	16.	1401.	19.21	87.63	3500.	8856.
185.	61747.	17.	1456.	18.29	87.63	3500.	8799.
190.	64021.	18.	1512.	17.41	87.63	3500.	8742.
195.	66270.	20.	1569.	16.56	87.63	3500.	8685.
200.	68492.	21.	1627.	15.74	87.63	3500.	8627.
205.	70683.	23.	1686.	14.96	87.63	3500.	8570.
210.	72842.	24.	1745.	14.21	87.63	3500.	8513.
215.	74964.	26.	1805.	13.48	87.63	3500.	8456.
220.	77048.	27.	1866.	12.78	87.63	3500.	8399.
225.	79092.	29.	1927.	12.11	87.63	3500.	8341.
230.	81093.	31.	1990.	11.47	87.63	3500.	8284.
235.	83049.	32.	2053.	10.85	87.63	3500.	8227.
240.	84958.	34.	2117.	10.26	87.63	3500.	8170.
245.	86819.	36.	2182.	9.69	87.63	3500.	8113.
250.	88629.	38.	2247.	9.14	87.63	3500.	8056.
255.	90388.	40.	2313.	8.61	87.63	3500.	7998.
260.	92093.	42.	2380.	8.11	87.63	3500.	7941.
265.	93744.	44.	2447.	7.62	87.63	3500.	7884.
270.	95340.	47.	2516.	7.16	87.63	3500.	7827.
275.	96879.	49.	2585.	6.71	87.63	3500.	7770.
280.	98360.	51.	2655.	6.28	87.63	3500.	7712.
285.	99784.	54.	2725.	5.87	87.63	3500.	7655.
290.	101148.	56.	2796.	5.48	87.63	3500.	7598.
295.	102454.	59.	2868.	5.10	87.63	3500.	7541.
300.	103700.	61.	2941.	4.74	87.63	3500.	7484.
305.	104887.	64.	3014.	4.40	87.63	3500.	7426.
310.	106013.	67.	3089.	4.07	87.63	3500.	7369.
315.	107080.	70.	3163.	3.76	87.63	3500.	7312.
320.	108088.	72.	3239.	3.46	87.63	3500.	7255.

325.	109036.	75.	3315.	3.18	87.63	3500.	7198.
330.	109926.	78.	3392.	2.91	87.63	3500.	7140.
335.	110758.	81.	3470.	2.65	87.63	3500.	7083.
340.	111532.	85.	3549.	2.41	87.63	3500.	7026.
345.	112250.	88.	3628.	2.18	87.63	3500.	6969.
350.	112912.	91.	3708.	1.96	87.63	3500.	6912.
355.	113520.	95.	3789.	1.76	87.63	3500.	6855.
360.	114075.	98.	3871.	1.57	87.63	3500.	6797.
365.	114579.	101.	3953.	1.39	87.63	3500.	6740.
370.	115032.	105.	4036.	1.22	87.63	3500.	6683.
375.	115438.	109.	4120.	1.06	87.63	3500.	6626.
380.	115797.	112.	4205.	0.92	87.63	3500.	6569.
385.	116111.	116.	4291.	0.78	87.63	3500.	6511.
390.	116384.	120.	4377.	0.66	87.63	3500.	6454.
395.	116616.	124.	4464.	0.55	87.63	3500.	6397.
400.	116812.	128.	4553.	0.45	87.63	3500.	6340.
405.	116973.	132.	4641.	0.36	87.63	3500.	6283.
410.	117102.	137.	4731.	0.28	87.63	3500.	6225.
415.	117203.	141.	4822.	0.21	87.63	3500.	6168.
420.	117278.	145.	4913.	0.15	87.63	3500.	6111.
425.	117331.	150.	5006.	0.10	87.63	3500.	6054.
430.	117365.	154.	5099.	0.06	87.63	3500.	5997.
435.	117384.	159.	5193.	0.03	87.63	3500.	5940.
440.	117393.	164.	5288.	0.01	87.63	3500.	5882.
445.	117395.	168.	5384.	0.00	87.63	3500.	5825.

Ideal Performance Delta Velocity is: 6254.70 <ft/s>

Boost Orbit:

Apocynthion	--	36.6000	<nm>
Pericynthion	--	19.3000	<nm>
Inclination	--	26.2500	<deg>
Longitude of the Ascending Node	--	283.0000	<deg>
Argument of Pericynthion	--	91.2600	<deg>
Eccentricity	--	0.0089	<nd>

Velocity Required at Apocynthion to Achieve
the Holding Orbit (37. X 50.) : 42.50 <ft/s>

Fuel Required for the Apocynthion Burn : 24.94 <lbm>

Weight After Apocynthion Burn : 5754.00 <lbm>

Weight of the Payload Placed in Orbit : 428.00 <lbm>

************ SIMULATION COMPLETE ***************